University of Alberta

Music Analysis/Synthesis by Optimized Multiple Wavetable Interpolation

by

Jonathan Jeffrey James Mohr

©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2002

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-81244-8

Canada

University of Alberta

Library Release Form

Name of Author: Jonathan Jeffrey James Mohr

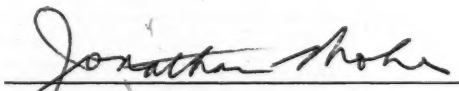
Title of Thesis: Music Analysis/Synthesis by Optimized Multiple Wavetable Interpolation

Degree: Doctor of Philosophy

Year this Degree Granted: 2002

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



Jonathan Jeffrey James Mohr
1 Woodridge Heights
Box 1974, Station Main
Camrose, Alberta
Canada, T4V 1X8

Date: 2002 October 1

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Music Analysis/Synthesis by Optimized Multiple Wavetable Interpolation** submitted by Jonathan Jeffrey James Mohr in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dr. Xiaobo Li
Supervisor

Dr. Michael Roeder

Dr. Anup Basu

Dr. Pawel Gburzynski

Dr. Zhi-Qiang Liu
External Examiner

Date: 2002 September 30

To James Neff, for encouraging me to study music;
to Chet Olson, for inviting me to study computing science;
and to Robyn, Joshua and Rebekah, for living with the consequences of both.

Abstract

Multiple wavetable interpolation is a form of music analysis/synthesis that involves three basic steps: 1) The recorded sound is reduced to a set of breakpoints by piecewise linear approximation of the spectral envelopes of its harmonics; 2) the spectrum at each breakpoint is matched by determining weightings for a small number of wavetables; and 3) the sound is resynthesized using multiple wavetable additive synthesis by interpolating between the weightings for each wavetable at consecutive breakpoints.

This thesis presents a new analysis/synthesis method, *optimized multiple wavetable interpolation*, that generalizes and optimizes multiple wavetable interpolation. The method uses a clustering algorithm to select a bank of wavetables such that the wavetables will be useful in matching the breakpoint spectra of a wide variety of harmonic tones played by various instruments. The breakpoint-matching algorithm selects subsets of the wavetables in the wavetable bank that best match each breakpoint spectrum, subject to the constraint that a wavetable that ceases to be used at a given breakpoint must be faded out by the next breakpoint and one that comes into use must be faded in. This algorithm introduces the use of the single-source acyclic weighted shortest path algorithm to choose breakpoint matches in a globally optimal way. The output of the algorithm is a sequence of n-tuples of pairs of wavetable indices and weights which can serve as a control stream for a hardware or software synthesizer.

A secondary contribution of this research is a new breakpoint-selection algorithm which operates by segment merging; the algorithm has characteristics that make it well suited to use on instrumental tones, especially those with vibrato or other pronounced amplitude changes.

Acknowledgements

The author wishes to acknowledge and express his sincere thanks to Augustana University College, Camrose, Alberta, for supporting his graduate studies in computing science, and to the National Sciences and Engineering Research Council of Canada for support through a Postgraduate Scholarship.

Thanks also to Dr. Xiaobo Li, a patient, supportive, and helpful advisor; to the other members of my graduate committee; to Dr. Lorna Stewart, Associate Chair (Graduate), for her work on my behalf; to Edith Drummond, the Graduate Program Coordinator, for all her assistance and advice; and to Dr. Tony Marsland for his advice and encouragement.

Acknowledgement and thanks are also due to the creators and maintainers of the software packages which were used in this research: to Dr. James Beauchamp and the members of the Computer Music Project at the University of Illinois at Urbana-Champaign for the *SNDAN* sound analysis package; to Barry Vercoe of the MIT Media Lab and the many other contributors to the music synthesis program `Csound`; to Dr. Diane Cook and Joseph Potts of the University of Texas at Arlington and Will Taylor of NASA for `AutoClass C`, based on the clustering program AutoClass III by Peter Cheeseman of RIACS and John Stutz of NASA; to Matthew Wall of MIT for `GAlib`: *A C++ Genetic Algorithm Library*; to Heiko Eissfeldt for `cdda2wav`; and to Thomas Eschenbacher for `kwave`.

Contents

1 Introduction **1**
- 1.1 Fundamentals of Music Analysis and Synthesis 2
  - 1.1.1 Definitions 3
  - 1.1.2 Musical Pitch Names 9
  - 1.1.3 Basic Synthesis Techniques 10
- 1.2 Optimized Multiple Wavetable Interpolation 15
  - 1.2.1 Purpose 15
  - 1.2.2 Indexed Color: An Analogy 17
  - 1.2.3 The New Analysis/Synthesis Method: The Analogy Applied 18
  - 1.2.4 Contribution 19

2 Music Analysis/Synthesis: An Overview **21**
- 2.1 Definition of Analysis/Synthesis 21
- 2.2 Fourier Analysis 22
  - 2.2.1 The Short-Time Fourier Transform (STFT) 23
- 2.3 Data Reduction 24
  - 2.3.1 Goals of Data Reduction 24
  - 2.3.2 Effects of Data Reduction 25
- 2.4 Methods of Analysis/Synthesis 28
  - 2.4.1 Phase Vocoder Analysis/Synthesis 28
  - 2.4.2 Peak-Tracking Phase Vocoder 28
  - 2.4.3 Karhunen-Loève Synthesis 29
  - 2.4.4 Spectral Modeling Synthesis 30
  - 2.4.5 Spectral Interpolation Synthesis 30
  - 2.4.6 Analysis-by-Synthesis/Overlap-Add 32
  - 2.4.7 Multiple Wavetable Synthesis 32
  - 2.4.8 Bandwidth-Enhanced Additive Modeling 34
  - 2.4.9 Multiple Wavetable Interpolation 34

3 Optimized Multiple Wavetable Interpolation **36**
- 3.1 Overview of the Process 36
- 3.2 The Breakpoint Matching Algorithm 45
  - 3.2.1 Match Evaluation 45
  - 3.2.2 The Initial Match 49

3.2.3 Overlapping of Wavetable Sets 53
3.2.4 Optimization of Oscillator Assignments 55

4 Implementation Issues **61**
4.1 Waveform Analysis 61
4.2 Breakpoint Selection 62
4.2.1 A Segment-Merging Algorithm 62
4.2.2 Error Measures 66
4.2.3 Number of Breakpoints 71
4.3 Wavetable Bank Selection 71
4.3.1 Use of a Clustering Algorithm 71
4.3.2 Grouping of Sample Tones by Pitch 74
4.3.3 Selection of Class Representatives 76
4.4 Breakpoint Matching 77
4.4.1 Implementation of Multi-Level Search 77
4.4.2 Implementation of the Genetic Algorithm 79
4.4.3 Implementation of Overlapping 81
4.4.4 Implementation of the Optimizer 82

5 Experimental Results **93**
5.1 Tones Selected for Testing 93
5.2 Results of Waveform Analysis 96
5.3 Breakpoint Selection Results 100
5.4 Wavetable Bank Selection Results 109
5.4.1 Clustering Results 109
5.4.2 Hand-Tuning of Wavetable Banks 114
5.5 Breakpoint Matching Results 117
5.5.1 Multi-Level Exhaustive Search Results 117
5.5.2 Genetic Algorithm Search Results 120
5.5.3 Oscillator Assignment Optimization 122
5.5.4 Optimization of Matches Found with a GA Search . . . 132
5.5.5 Comparison with Horner's Method 134
5.5.6 Data Reduction 137
5.6 Synthesis Results 138

6 Conclusions and Suggestions for Further Research **140**

Bibliography **144**

A Search Results in Detail **152**

B Optimization Results in Detail **160**

List of Figures

1.1 The first 100 ms of a French horn waveform. 3
1.2 A single cycle from a French horn waveform. 4
1.3 The ADSR amplitude envelope. 6
1.4 Amplitude envelopes which do not accord well with the prototypical ADSR envelope. 7
1.5 The spectral envelope of a trumpet tone. 8
1.6 The octave naming scheme. 11
1.7 Table scanning for table-lookup synthesis. 12

2.1 Three spectral ramps. 31

3.1 The stages of optimized multiple wavetable interpolation analysis/synthesis. 37
3.2 Phase vocoder analysis of a French horn playing pitch G3. . . 38
3.3 Phase vocoder analysis of a saxophone playing pitch G3. . . . 38
3.4 Piecewise linear approximation of a French horn playing pitch G3. 40
3.5 Piecewise linear approximation of a saxophone playing pitch G3. 40
3.6 Possible oscillator assignments with four oscillators. 43
3.7 The optimized multiple wavetable interpolation synthesis model. 46
3.8 The stages of the globally optimal breakpoint matching algorithm. 47
3.9 Search tree pruning by a "2+1" multi-level search. 51
3.10 A possible sequence of wavetable matches and two possible optimized oscillator assignments. 53
3.11 Other possible optimized oscillator assignments. 54
3.12 Analysis of a French horn tone at pitch E2. 56
3.13 Two-breakpoint overlapping with re-augmentation for a four-oscillator match of a French horn playing pitch E2. 57
3.14 Oscillator assignment for a four-oscillator match of a French horn playing pitch E2. 59

4.1 Fifty-five breakpoint PLA's of a violin tone at pitch A4 found by a genetic algorithm and by segment merging. 64
4.2 Comparison of PLA's of different resolution of a violin tone at pitch A4 found by segment merging. 65
4.3 Relative vs. absolute error in PLA's of a violin tone at pitch A4. 68

4.4 Relative vs. absolute error in 24-breakpoint PLA's of a bassoon tone at pitch C#2. 69
4.5 PLA of a flute chiff. 72
4.6 Low-amplitude vibrato in an English horn tone at pitch G3. . 73
4.7 Pitch classes selected for testing. 75
4.8 Initial and augmented matches to the breakpoints of a French horn playing pitch E2. 78
4.9 Procedure `GenerateEdges`. 85
4.10 Procedure `AddSubsets`. 86
4.11 Procedure `ReplaceZeros`. 87
4.12 Procedure `AddVertexAndEdge`. 88
4.13 Call diagram for `GenerateEdges` and related procedures. . . . 89

5.1 Analysis of a trumpet G5 tone. 97
5.2 Analysis of a glockenspiel G5 tone. 98
5.3 Analysis of a glockenspiel G5 tone using an analysis frequency of 41 Hz. 99
5.4 Inharmonicity of the upper partials of a piano tone. 101
5.5 Analysis and 20-breakpoint PLA of a clarinet A♯4 tone. 105
5.6 Comparison of two methods to model the initial peak of a clarinet A♯4 tone. 106
5.7 Phase vocoder analysis and 24-breakpoint PLA of a piano G3 tone. 107
5.8 Phase vocoder analysis and 24-breakpoint PLA of a piano G3 tone (two-dimensional view). 108
5.9 Analysis and piecewise-linear approximation of a viola G3 tone. 112
5.10 Graph of multi-level exhaustive search results. 121
5.11 Graph of GA search results compared with multi-level exhaustive search for Group 1. 122
5.12 Graphs of 3-oscillator optimization results. 128
5.13 Graphs of 4-oscillator optimization results. 129
5.14 Graphs of 5-oscillator optimization results. 130
5.15 Graph of GA optimization results compared with 3-, 4-, and 5-oscillator optimizations of multi-level exhaustive searches for Group 1. 133
5.16 Comparison of Horner's constrained matching with optimized multi-level exhaustive search results for Group 1. 135
5.17 Oscillator assignment resulting from Horner's method for a four-oscillator match of a French horn E2. 136

List of Tables

1.1 The standard frequencies of equal-tempered musical pitches. . 11

2.1 Categories of musical sound differentiability. 25

5.1 Durations and grouping of selected instrument tones. 94
5.2 Number of harmonics retained in the basis spectra for each group of tones. 95
5.3 Error of piecewise-linear approximations of non-vibrato tones. 102
5.4 Error of piecewise-linear approximations of tones with vibrato. 104
5.5 Duration and weight of a frame-weighted attack bias applied to selected flute tones. 109
5.6 Number of classes of breakpoints in each group. 110
5.7 The classes to which the breakpoint spectra of four example tones are assigned by clustering. 111
5.8 The classes to which spectra from each tone in Group 1 were assigned by clustering. 113
5.9 Number of basis spectra in each wavetable bank. 116
5.10 Summary of multi-level exhaustive search results. 120
5.11 Mean and maximum sizes of wavetable sets created by overlapping. 124
5.12 Summary of 3-, 4-, and 5-oscillator optimization results for various initial search types. 127
5.13 Mean matching error levels achieved by selecting the best match to each tone from among the optimizations of multiple different initial search strategies. 132
5.14 Sizes of wavetable envelope files. 137

A.1 Group 1, one-level exhaustive search. 153
A.2 Group 1, two-level exhaustive search with 1-table augmentation. 154
A.3 Group 1, two-level exhaustive search with 2-table augmentation. 155
A.4 Genetic algorithm search on Group 1, with population size of 100 and termination on convergence after 50 generations (thorough version). 156
A.5 Augmented genetic algorithm search on Group 1, thorough version. 157

A.6 Genetic algorithm search on Group 1, with population size of 50 and termination on convergence after 25 generations (quick version). 158
A.7 Augmented genetic algorithm search on Group 1, quick version. 159

B.1 Group 1, 3-oscillator, 3-table match optimization. 161
B.2 Group 1, 3-oscillator, 2- and 1-table match optimization. . . . 162
B.3 Group 1, 4-oscillator, 4-table match optimization. 163
B.4 Group 1, 4-oscillator, 3-table match optimization. 164
B.5 Group 1, 5-oscillator, 5-table match optimization. 165
B.6 Group 1, 5-oscillator, 4-table match optimization. 166
B.7 Optimization time and RMS error of Group 1, 5-oscillator, 4-table matches with limits on overlapping and wavetable set size. 167
B.8 Numbers of vertices and edges in optimization graphs. 168
B.9 Optimization of matches found by augmented thorough GA search. 169
B.10 Optimization of matches found by augmented quick GA search. 170
B.11 Group 1, Horner's constrained matching results. 171

Chapter 1

Introduction

"Computer music" appears to be an oxymoron, a juxtaposition of opposites. The computer represents, to most contemporary minds, the unfeeling machine, the hardware of mindless logic, the cross-product of mathematics and engineering. Music, on the other hand, is the most abstract and ineffable of the arts: its medium is not the charcoal of drawing, the stone of sculpting, the glass, steel, and wood of architecture, nor even the shapes and movements of human bodies that give dance its corporeality, but only waves—compressions and rarefactions—in the air; its canvas is not paper, nor cloth, nor a stage, but time alone; its frame, silence.

Yet, from the earliest days of the computer, musicians have been attracted to the possibilities offered by this programmable machine. In fact, long before the first electronic computer was constructed, composers were seeking the generality which sound synthesis by computer would eventually offer:

> I dream of instruments obedient to my thought and which, with their contribution to a whole new world of unsuspected sounds, will lend themselves to the exigencies of my inner rhythm.
>
> *Edgard Varèse* (1917) [69, page 141]

While acknowledging that computer music required musicians "to gain competence in areas which are seemingly foreign to music," John Chowning suggests that "it is not surprising that composers were the first artists to make substantive use of computers" since "programming involves mental processes and rigorous attention to detail not unlike those involved in composition" [15, page ix].

Hubert Howe, another early user of the computer for music composition, sees the attraction of the computer as its ability to support creativity by allowing new ideas to be tested. In the area of sound generation, Howe points out that the limitless possibilities of computer synthesis—"*any* describable sound can be produced"—are in fact limited by human factors—"the limitation is not in the capability of the computer but rather in the ability of composers to provide adequate descriptions of what they want"—and that this limitation

leads to creativity: "composers are encouraged to be creative with the qualities of the sounds they produce, by the very procedure by which they must work" [46, p. 166].

F. Richard Moore resolves the apparent contradictions in the term "computer music" by viewing this new field of study as strongly interdisciplinary, placing computer music at the center of a disciplinary context including music (theory, composition, and performance), computer science (programming), engineering (digital signal processing), physics (acoustics), and psychology (cognition and psychoacoustics) [65, page 24].

Two of the earliest uses of the computer in musical contexts were for the analysis [76, 77] and synthesis [80, p. 87] of musical tones, and they remain important applications in computer music [80, pp. 495–496]. Used in combination, analysis and synthesis allow the generation of musical tones (and, more generally, other sounds, including speech) from analysis data which has typically been data-reduced and may have been modified (as discussed in Chapter 2). Analysis/synthesis algorithms have been used in a variety of forms and contexts, from incorporation in the hardware of consumer-oriented synthesizers ("keyboards") [80, pp. 159–160] to use in multimedia software, from experiments with data reduction [7, 13, 33, 54, 59, 76, 83, 90] to experiments on the perception of musical tones [3, 30, 39, 40, 51, 55], from producing sound effects for games to enabling creative artistic expression in musical compositions [80, pp. 146–148].

This thesis presents a new analysis/synthesis method, called *optimized multiple wavetable interpolation*, which is based on multiple wavetable interpolation, an efficient form of additive synthesis. These terms and concepts will be more fully defined in Chapter 2, but a brief overview and an analogy may help to provide a context for and orientation to the subject matter. Readers who are familiar with the principles of music analysis and synthesis may wish to omit §1.1 and proceed to the description of the proposed new method in §1.2.

1.1 Fundamentals of Music Analysis and Synthesis

Analysis/synthesis is a general process in which a recorded sound is analyzed in such a way that a musician or sound technician can modify the analysis data and synthesize an altered sound from the modified data; alternatively, the goal of the analysis may be to find a data-reduced representation from which the original tone may be resynthesized economically but with high fidelity, typically by a commercial synthesizer ("keyboard").[1]

The focus of the present research was on music analysis/synthesis, in partic-

[1] Analysis/synthesis may be used toward other goals as well, such as to gain insight regarding the perceptual features of the sound [78, 96], but the two goals listed here are the primary goals toward which this research is directed.

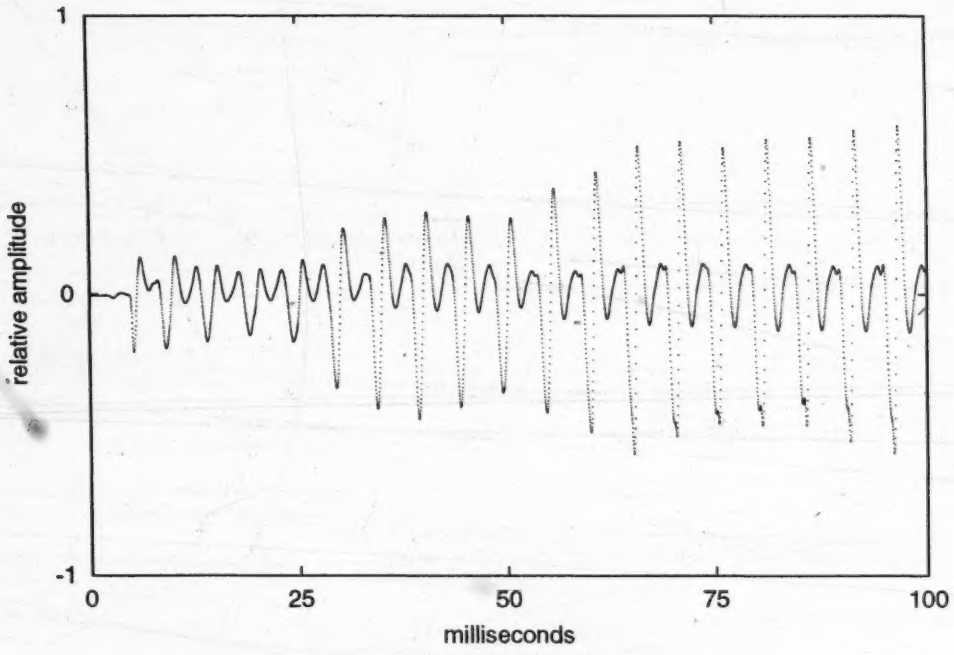


Figure 1.1: The first 100 ms of a French horn waveform.

ular, on the analysis and synthesis of isolated monophonic instrumental tones (i.e., single notes played by individual instruments). It may be useful for musicians and non-musicians alike to review some of the terms and concepts of signal processing in a musical context.

1.1.1 Definitions

A *note* is the notation for a single sound event; when a singer sings that note or an instrumentalist plays it, the audible result is a *tone.* If a tone is considered from a physical point of view, it may be referred to as a *signal.* If we wish to emphasize the "shape" of a signal as it might appear on an oscilloscope or a graph, we might refer to it as a *waveform*; Figure 1.1 shows the first 100 ms of a French horn waveform. The oscilloscope display might be driven by an analog (continuous) input signal; the graph of a waveform would more likely have been generated by *sampling* an analog signal to produce its digital approximation. Audio signals are sampled at a rate of 44100 samples per second for compact disk (CD) recordings and at 48000 samples per second for digital audio tape (DAT). Each *sample* is represented by a dot in Figure 1.1.

The term *waveform* is also used with reference to a single *cycle* of a signal; Figure 1.2 shows a single cycle from the middle of the same horn tone, a duration of about 5 ms. The relative position of each point in the cycle is referred to as its *phase*; the phase axis of the figure is labelled in radians

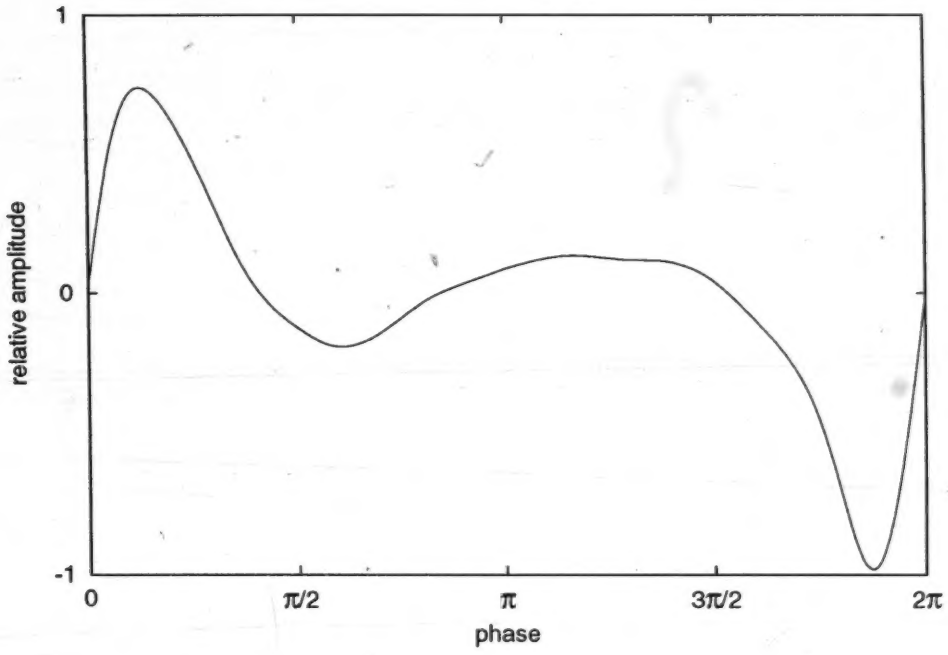


Figure 1.2: A single cycle from a French horn waveform.

$(0 \ldots 2\pi)$. The duration of a single cycle of an oscillation is its *period.* A signal which consists primarily of repetitions of the same single-cycle waveform is said to be *periodic*; most musical tones are quasi-periodic, since the shape of the waveform tends to remain the same or change slowly as the tone is sustained. A signal which is random or quasi-random is categorized as *noise.*

A tone has several characteristics, which may also be described from a physical or a perceptual point of view:

Amplitude/Loudness The *amplitude* of the audio signal—the degree to which the air is compressed and rarefied by the sound generator—is perceived (non-linearly) as *loudness*, the quality which is controlled by the "volume" control on an amplifier.

The amplitude of a signal typically changes over time: the rising and falling shape of a signal's amplitude over time is called its *amplitude envelope.* Because the tones of many musical instruments begin with an *attack*—a sudden increase in amplitude—then quickly fall (due to the *decay* of some initial transients) to a *sustained* level for the main body of the tone (also called the "steady state") until the final *release*, authors in the field of computer and electronic music often refer to a prototypical ADSR envelope, as illustrated in part (a) of Figure 1.3.[2] The amplitude

[2]The ADSR envelope is a generalized simplification which was used on analog synthesizers

envelope of a trumpet tone illustrated in part (b) of Figure 1.3 conforms fairly well to the model ADSR envelope; the envelopes of other tones, especially those with *vibrato*—a slow fluctuation in frequency for the purpose of giving warmth or richness to the tone—such as the violin tone of Figure 1.4, part (a),[3] and those of percussive instruments which cannot sustain tones, such as the piano tone shown in part (b) of the same figure, accord less well with the simple linear model.

Frequency/Pitch The *frequency* of a signal—the rate of the oscillation of the sound generator, typically measured in cycles per second —is perceived (in general) as the *pitch* of the tone, the relative location of the tone on a scale of "high" to "low". A shorter, thinner, and/or tighter string will vibrate more quickly—and thus produce a higher pitch—than a longer, thicker, looser string.

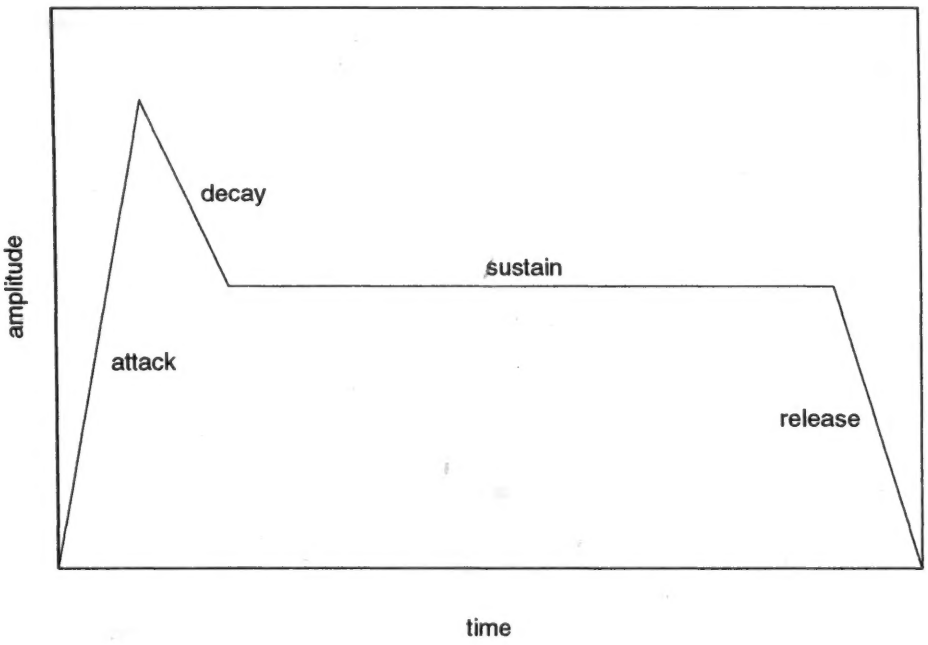
This definition is an oversimplification, because almost no natural oscillator (as opposed to a mechanical, electrical, or electronic oscillator, such as a sine wave generator) vibrates at a single frequency. For example, a violin string vibrates simultaneously across its entire length, in two halves, in three thirds, and so on, and may also vibrate in irregular ways, especially when it is just starting to vibrate due to being struck, pulled, or bowed. For such a tone, it is more precise to refer to its *fundamental frequency*, the frequency of the lowest or most basic mode of vibration of the oscillator;[4] the perceived pitch of a tone is primarily determined by its fundamental frequency.

Spectrum/Timbre If an oscillator is vibrating simultaneously in several modes (e.g., in full, in halves, in thirds, etc.), its waveform can be resolved into its constituent components, each of which is a sine wave at a single frequency; this analysis is typically performed by some form of Fourier analysis, as discussed in §2.2. These components are collectively referred to as the *spectrum* of a periodic signal; individually, each component is called a *partial*. Just as the waveform shows the various energy levels of a signal over time, the spectrum represents its energy

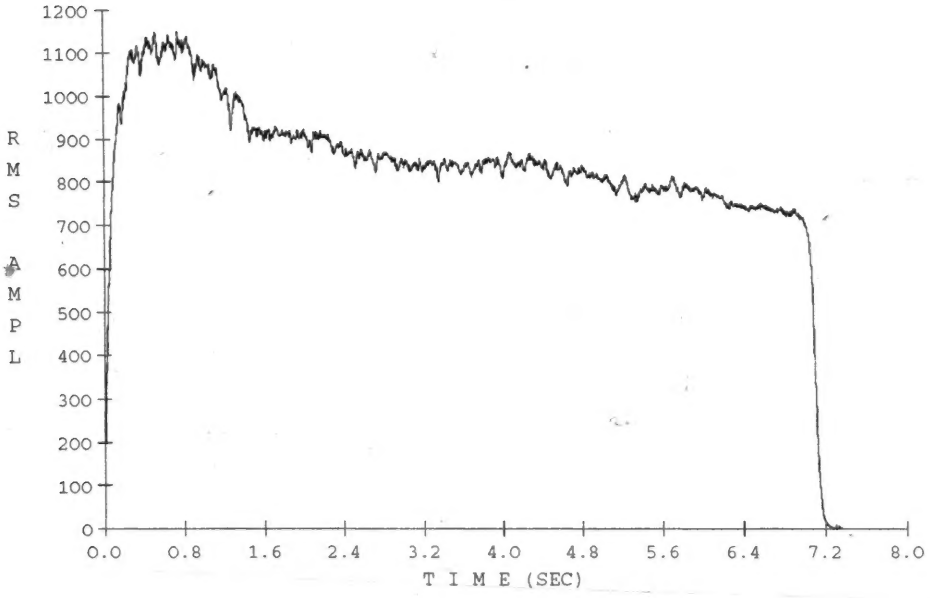
to define amplitude envelopes [80, p. 97]. Several authors use a simpler three-segment "attack-sustain-release" or "attack-sustain-decay" envelope as a prototype instead of or in addition to the ADSR model [20, 66]; Jensen recently proposed an ASR envelope with an inflection point in each of the attack and release segments as the basis of his timbre model [51].

[3]The fluctuations seen in Figure 1.4(a) would more properly be called *tremolo*, since they are fluctuations in amplitude, not in frequency; however, this is an example of vibrato-induced tremolo [31], since it is a side effect of the back-and-forth movement of the violinist's finger on the string, which causes both amplitude and frequency fluctuation, the latter being more perceptible than the former [75]. The trumpet tone exhibits *shimmer* [51]—small, rapid amplitude fluctuations.

[4]The concept of a fundamental frequency does not apply to all instruments; for example, a chime is characterized by a "strike tone" and a "hum tone" [6, 79].

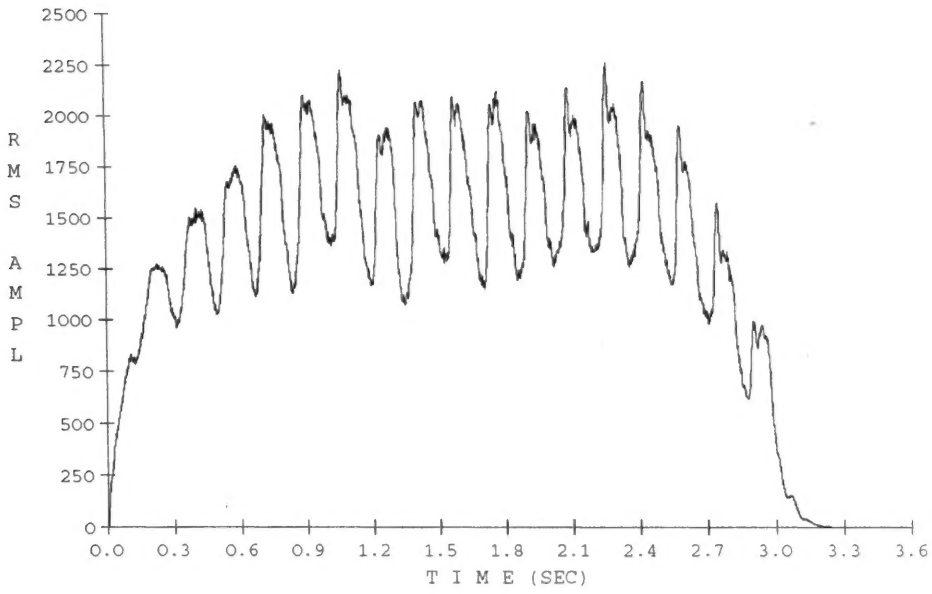


(a) The prototypical ADSR amplitude envelope.

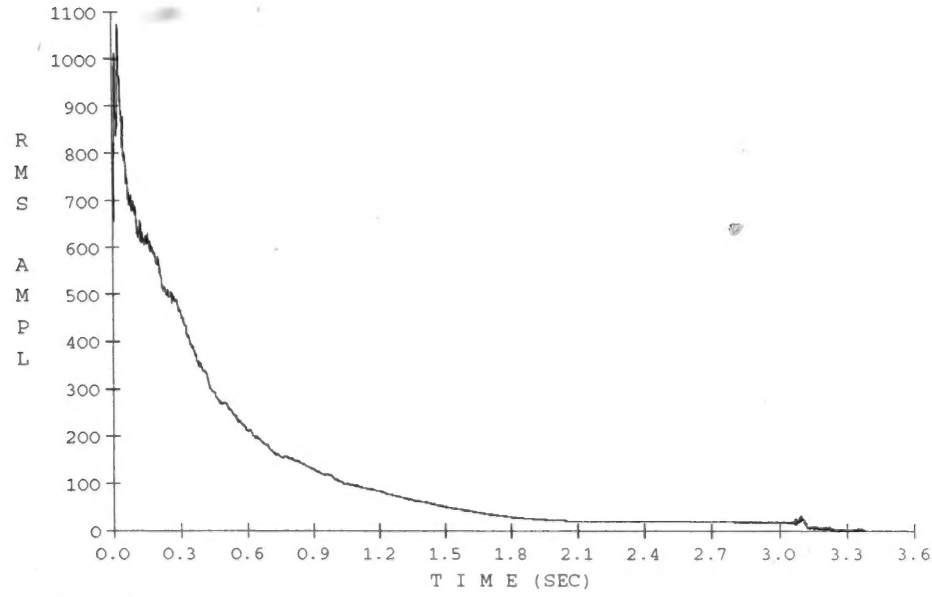


(b) The ADSR-like amplitude envelope of a trumpet tone.

Figure 1.3: The ADSR amplitude envelope.



(a) The amplitude envelope of a violin tone with vibrato.



(b) The amplitude envelope of a piano tone.

Figure 1.4: Amplitude envelopes which do not accord well with the prototypical ADSR envelope.

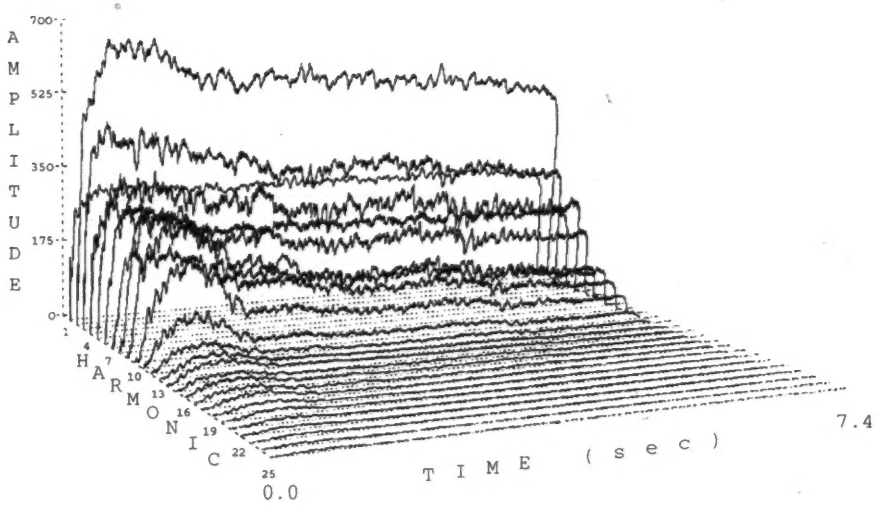


Figure 1.5: The spectral envelope of a trumpet tone.

distribution in the frequency domain. Since each partial can vary in amplitude over time independently of the other partials in a quasi-periodic tone, we can speak of the *spectral envelope* of a complex signal, which is the surface formed by the changes in spectral shape over time;[5] Figure 1.5 shows the spectral envelope of the same trumpet tone for which the amplitude envelope was shown in part (b) of Figure 1.3.

Harmonics are sinusoidal components (partials) with frequencies that are integral multiples of the fundamental. To return to the example of a string vibrating simultaneously in multiple segments, the two halves of the string vibrate at a frequency twice that of the whole string, the three thirds, at three times the fundamental frequency, and so on. Partials above the fundamental are commonly called *overtones*. If all the overtones are harmonics (or nearly so), the waveform is called *harmonic*; otherwise, it is *inharmonic* or *aperiodic*. The word "tone" is usually used with reference to a harmonic sound or at least a quasi-harmonic sound with a perceptible pitch, such as the gong of a bell, which includes some inharmonic partials.

Timbre is the perceptual counterpart of the spectrum of a tone; along

[5] Risset and Wessel refer to the profile of spectral evolution rather poetically as a "spectromorphological gait" [79].

with other perceptual cues[6] such as the nature of the attack,[7] timbre allows us to identify the sounds produced by different instruments, to recognize a loved one's voice, and to distinguish between a gunshot and a car backfiring (usually).

Fourier analysis of a signal yields an amplitude and a phase (from which the instantaneous frequency can be calculated) for each sinusoidal component. The amplitude of the whole signal (i.e., of all the partials of the spectrum) is often expressed in terms of a root-mean-square (RMS) amplitude: the square root of the mean of the squares of the amplitudes of each partial.

$$\underset{\text{Amplitude}}{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(a_k)^2} \tag{1.1}$$

The signal amplitudes shown in Figures 1.3 and 1.4 were calculated as RMS amplitudes.

1.1.2 Musical Pitch Names

Although many musical instruments can produce a tone at any desired frequency in their respective ranges, others (such as the piano) can produce tones at only selected frequencies, and conventional Western music restricts itself to only those pitches for the most part, even when played on instruments with continuous ranges.

The distance between any two pitches is called an *interval.* The interval between a fundamental and its first overtone, with a frequency ratio of 1:2, is the most basic or salient interval, at least in a perceptual and structural sense: a pitch and its frequency doubling are heard as being closely related, and will fuse into a single sonority if played together. This interval is the *octave*;[8] it is divided into twelve equal[9] intervals called *semitones*, the smallest

[6] Psychoacousticians distinguish between timbre—the perceptual manifestation of the combined partials of a complex tone at a given instant—and *sonance*—the transient attributes of a tone: attack and decay characteristics of partial tones, fusion, phase differences, and other aperiodic attributes of the sound [95, p. 29][82, citing [85]]; most authors in the field of computer music, however, use the term "timbre" in a broader sense.

[7] It is well known that transients carry important information for the identification of traditional instruments, particularly as they characterize the attack portion of a musical sound [8, 82].

[8] The octave is so named because the earliest Western music used only seven different pitches within that most fundamental interval, so the pitch at the upper end of the interval, completing the *scale*, was the eighth one. Medievals would have seen significance in the fact that there were also seven days in the week: the Sunday after Easter Sunday was called the Octave of Easter.

[9] The octave is divided into equal intervals only in the tuning system called Equal Temperament, in which the frequency ratio of a twelfth of an octave is $\sqrt[12]{2}$. Historically and in practice today, many other tuning systems have been used, in which the smallest intervals

available intervals on a standard piano keyboard. Pitches within each octave are named using seven different letters—A to G—and a few modifying symbols—primarily '♯' (sharp) and '♭' (flat);[10] all the pitches which are at the same position in each octave (for example, all the E♭'s) are called a *pitch class*.

There are various extant systems for naming the octaves of the standard musical gamut; for historical reasons, they typically (and illogically) begin each new octave at the pitch class C, not A. The system used here is a simple one: each octave is numbered, beginning with octave 0, which contains pitches at frequencies at or below the lower limit of human hearing;[11] this octave starts at C0, the pitch of the largest pipe of a "32-foot" stop (group of pipes) on a pipe organ, and includes the lowest pitch on the piano, A0. The pitch of the lowest note of a double bass with a C-extension, C1, starts the next octave, which also contains the lowest bassoon, bass saxophone, contrabass clarinet, and tuba tones. Octave 4 starts at "middle C", which is about in the middle of the piano keyboard, and includes A4, the international pitch standard, at 440 Hz; the standard frequency of all other pitches (as shown in Table 1.1) can be calculated from this value using the formula

$$\text{frequency}_p = 440 \cdot 2^{n/12} \tag{1.2}$$

where frequency_p is the standard frequency of pitch p and n is the number of semitones from A4 to p (positive if p is higher than A4, negative if lower). Octave 8 includes the highest pitches produced by standard orchestral instruments, including C8, the pitch of the top key on the piano, the top bar on both the xylophone and the glockenspiel, and the highest piccolo tone.[12] This naming scheme is illustrated in Figure 1.6.

1.1.3 Basic Synthesis Techniques

Additive Synthesis

Fourier's theorem posits that any periodic signal that meets the Dirichlet conditions[13] can be expressed as the sum of a (possibly infinite) number of

are almost but not exactly equal. Most of these systems resulted from the attempt to tune perfectly both octaves and the second-most-important interval—that between the first and second overtones, with a ratio of 2:3—even though the ratios 2:1 and 3:2 are incommensurable: there are no integers m and n such that $2^m = (3/2)^n$.

[10]Some combinations of letters and symbols form pitch names which are synonyms for other pitch names; such names are called *enharmonic equivalents*. For example, A♯ and B♭ are alternative names for the same pitch (at least on a piano keyboard—some string players inflect them slightly differently). Which name is the correct one to use depends on the notational context; since pitch names are used in a context-independent way in this thesis, the choice of one name over its enharmonic equivalents is arbitrary.

[11]The range of human hearing is about 16 Hz to 20,000 Hz; the range tends to contract from both extremes with age [85].

[12]To save space, the table does not include the frequency of C8, which is 4186.01.

[13]The signal must correspond to a piecewise regular function which has a finite number of finite discontinuities and a finite number of extrema; any signal generated by natural

Pitch Class	Octave 0	1	2	3	4	5	6	7
C	16.35	32.70	65.41	130.81	261.63	523.25	1046.5	2093.0
C♯/D♭	17.32	34.65	69.30	138.59	277.18	554.37	1108.7	2217.5
D	18.35	36.71	73.42	146.83	293.66	587.33	1174.7	2349.3
D♯/E♭	19.45	38.89	77.78	155.56	311.13	622.25	1244.5	2489.0
E	20.60	41.20	82.41	164.81	329.63	659.26	1318.5	2637.0
F	21.83	43.65	87.31	174.61	349.23	698.46	1396.9	2793.8
F♯/G♭	23.12	46.25	92.50	185.00	369.99	739.99	1480.0	2960.0
G	24.50	49.00	98.00	196.00	392.00	783.99	1568.0	3136.0
G♯/A♭	25.96	51.91	103.83	207.65	415.30	830.61	1661.2	3322.4
A	27.50	55.00	110.00	220.00	440.00	880.00	1760.0	3520.0
A♯/B♭	29.14	58.27	116.54	233.08	466.16	932.33	1864.7	3729.3
B	30.87	61.74	123.47	246.94	493.88	987.77	1975.5	3951.1

Table 1.1: The standard frequencies of equal-tempered musical pitches.

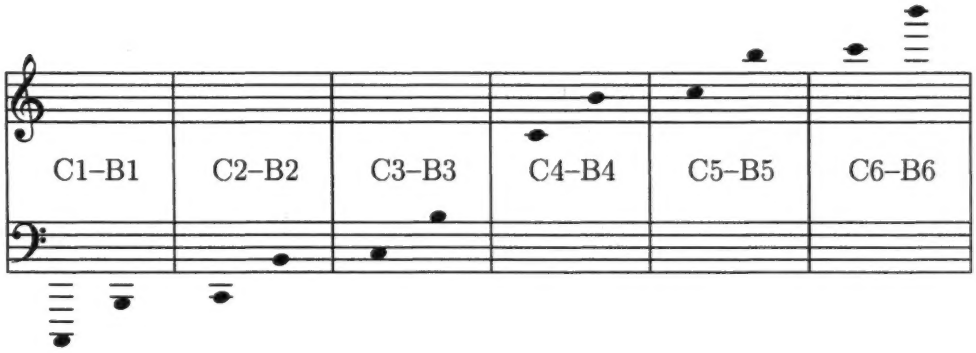


Figure 1.6: The octave naming scheme.

harmonically related sinusoids, each with a particular amplitude and phase. Thus, a complex waveform can be synthesized by adding together a number of sinusoids with independently varying amplitude, frequency, and phase.

The sinusoids can be generated by a bank of oscillators, but a more economical method is to store a single cycle of a discrete (sampled) sinusoidal waveform in a lookup table, called a *wavetable*. The phase of the desired sinusoid can then be used to determine the starting index in the table, and the desired frequency determines the step size to use in accessing the next sample to be looked up in the table, as illustrated in Figure 1.7. The increment by which the scanning (or *resampling*) process steps through the wavetable may

physical means, such as by a musical instrument, will satisfy these conditions [63].

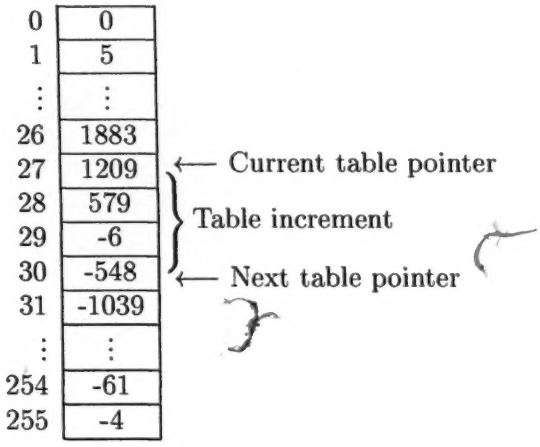


Figure 1.7: Table scanning for table-lookup synthesis.

be calculated as

$$\text{increment} = \frac{N \cdot F_{\text{output}}}{F_{\text{sampling}}} \tag{1.3}$$

where N is the table size, F_{output} is the desired output frequency, and F_{sampling} is the sampling frequency of the output file or device; for example, for CD-quality output ($F_{\text{sampling}} = 44100$ Hz) with a table size of 1024, an output signal at middle C ($F_{\text{output}} = 261.63$ Hz) would require an increment of 6.075. Since the index value computed from the previous index and the increment must be an integer, either the result must be truncated or rounded, or an interpolated value must be calculated from the two table values at $\lfloor \text{index} \rfloor$ and $\lceil \text{index} \rceil$; interpolation reduces the *table-lookup noise* [17, 64], but the additional cost can be avoided by using larger table sizes (such as 8192). The wavetable is treated as a circular array: if the current index plus the increment would access a point beyond the end of the wavetable, the index value wraps around to the start of the wavetable.

A *digital oscillator* generates a sound by repeatedly scanning a wavetable and sending the samples through a digital-to-analog converter (DAC); this process is called *table-lookup synthesis* [11][80, p. 90–96]. The outputs of several digital oscillators may be summed to form a composite sound waveform with a spectrum of arbitrary complexity.

If the frequency and amplitude of each sinusoid is held constant for the duration of the tone being synthesized, the desired timbre will have been generated, but this will approximate only the steady-state portion of an instrumental or naturally occurring sound. Most sounds vary in timbre from millisecond to millisecond, and in very complex fashion. In fact, each partial typically has its own unique amplitude and frequency profile (envelope) over time.

A complete additive synthesis instrument thus requires that the digital oscillators be controlled by separate amplitude and frequency envelope generators or *driving functions*. This control data could be specified by a composer, either manually or through the use of a composition program or an interactive performance system, or it could be determined by analyzing a natural sound and using the (possibly modified) amplitude and frequency parameters to synthesize a new tone [80, pages 143–144]. The latter method is referred to as *analysis/synthesis*.

The advantage of sinusoidal additive synthesis is that any desired waveform can be synthesized and the parameters of the algorithm are relatively intuitive: musicians are used to thinking of a complex tone as a fundamental tone with harmonically related overtones of various amplitudes. Theoretically, if a short-time Fourier transform is used for analysis and additive synthesis is used for resynthesis, as in the phase vocoder (see Section 2.4.1), the system has the identity property: it determines additive synthesis parameters from the samples of a waveform, and can exactly reproduce the waveform using those parameters [73].

The disadvantages are the large amount of control information that is needed to accurately resynthesize a desired tone and the large amount of computation required: it is often necessary to add thirty or more partials (sinusoids with frequencies which may or may not be integer multiples of a fundamental frequency) in order to achieve a desired degree of accuracy.

Wavetable Synthesis

The computational cost of additive synthesis can be greatly reduced by the simple observation that the waveform which is placed in a digital oscillator's lookup table need not be a pure sinusoid: it could be a single cycle of *any* periodic waveform. This variant of additive synthesis is called *wavetable synthesis*.[14] If, for example, the waveform of Figure 1.2 were loaded into the lookup table, the oscillator would generate the steady-state spectrum of a particular French horn tone; this would be accomplished using a single digital oscillator, as opposed to the 112 oscillators which would be required to accurately reproduce the same spectrum at CD quality using sinusoids.

Just as additive synthesis requires additional control data to synthesize a sound which changes over time, wavetable synthesis requires a higher-level control structure to model or create interesting or realistic tones. Of course, the frequency and/or amplitude of the tone could be modified by driving functions to control the oscillator; such modifications may suffice to model the sustain portion of a horn tone, but they cannot accurately recreate the spectral en-

[14] Wavetable synthesis should be distinguished from *sampling*, in which a large lookup table is loaded with a digital recording of an arbitrary sound of up to several seconds duration, possibly including all of the attack, sustain, and release portions of the sound, and the table is scanned only a single time (i.e., *not* in circular fashion) to play back the sound, shifted to any desired frequency by resampling.

velopes of the attack or release segments.

A simplistic solution would be to load different waveforms into the lookup table as the synthesis proceeds, but this method requires many changes of waveforms with only slight differences between successive spectra in order to avoid audible clicks due to perceptible discontinuities [86]. One of the earliest implementations of a digital synthesizer for microcomputers simply switched from one wavetable to another during the course of synthesis, but audible background noise was reported [11].

Several different methods have been proposed to allow the generation of changing spectra by wavetable synthesis, all of which use multiple wavetables:

Wavetable crossfading The synthesis proceeds by *crossfading* from one wavetable to another, fading one table out (i.e., gradually reducing its amplitude to zero) while another table is faded in. This approach has been implemented in many commercial synthesizers [80, pp. 159–160], but the user must generate the desired sounds by trial-and-error using physical controls or a "patch editor".

Wavetable interpolation As discussed more fully in §2.4.5, Serra, Rubine, and Dannenberg's *spectral interpolation synthesis* [86] proposes an analysis method which selects a number of *breakpoints* throughout the duration of the tone such that the original tone can be approximated by interpolating between the spectra at successive pairs of breakpoints: the spectrum at one breakpoint is loaded into one wavetable, the spectrum from the next breakpoint is loaded into a second wavetable, then the portion of the output signal corresponding to the segment between the two breakpoints is synthesized by adding the weighted outputs of the two digital oscillators together, where the weight of the first oscillator fades out from one to zero while the weight of the second fades in from zero to one. When the weight of the first oscillator reaches zero, the waveform in its table is changed to the spectrum of the breakpoint after next, and synthesis of the next segment proceeds with the fade in of the waveform in the first oscillator's wavetable and the fade out of the second.

Multiple wavetable synthesis Serra et al. also proposed a form of wavetable interpolation which used nonlinear weighting functions, an idea which was generalized by Horner [41, 44] as *multiple wavetable synthesis*, the weighted additive combination of several wavetables. The waveform stored in each oscillator's wavetable remains the same throughout the synthesis of a given tone; in order to approximate varying spectra over time, different weightings are used for each wavetable at each time unit of the synthesis. Horner's contribution, reviewed in §2.4.7, was to propose methods for the selection of the waveforms to be loaded into the wavetables and for determining the set of wavetable weightings at each point in time which will best approximate the spectrum of the original tone at that point.

Multiple wavetable interpolation Horner subsequently proposed [42] a generalization of wavetable interpolation which uses the concept of interpolating between the spectra of successive breakpoints, but allows the use of multiple weighted wavetables to match the spectrum at each breakpoint. In order to limit the number of oscillators used for synthesis, the proposed method, presented more fully in §2.4.9, uses only a subset of the selected waveforms (or *basis spectra*) to match the spectrum at each breakpoint, leaving one oscillator to fade out one of the wavetables from one breakpoint to the next, at which point a new waveform is loaded into the oscillator's wavetable to be faded in by the following breakpoint.

1.2 Optimized Multiple Wavetable Interpolation

Optimized multiple wavetable interpolation is a new analysis/synthesis method which seeks to extend and generalize multiple wavetable interpolation by allowing waveforms to be reused not only during the course of synthesizing a particular tone but in synthesizing a wide variety of different tones. Where Horner's multiple wavetable interpolation method selects "spectral snapshots" from various points within a given tone to be used in weighted subsets to match the breakpoint spectra of the same tone, optimized multiple wavetable interpolation selects a larger set of waveforms from tones played at selected pitches by many different instruments. These waveforms are then (conceptually) loaded into a bank of wavetables and used to synthesize multiple tones with a few oscillators. Instead of each oscillator having access to only its own dedicated wavetable into which different waveforms may be loaded at different times, a given oscillator accesses different wavetables from the bank during the course of synthesizing a tone, referring to each wavetable by its index in the wavetable bank.

1.2.1 Purpose

The motivation for this research was the observation that, while spectra from the steady-state portion of a given instrumental tone are likely to be similar to spectra from other tones played by the same instrument, it is also quite likely that a spectrum from one point of a tone played by one instrument is very similar to a spectrum from a different point of some tone played by a different instrument; furthermore, a given spectrum might be more similar to many other spectra from various points in multiple tones played by several different instruments than to other spectra from its own tone of origin. For example, a spectrum from the 15 ms mark of the attack portion of a particular bassoon tone might be more like a pair of spectra from the release of a bass clarinet tone, a waveform from a trombone attack, various points in different

piano tones, and a series of 'cello spectra than it is to any other spectrum in the same bassoon tone.[15]

The implications of this possibility are clear: By spending some additional time when designing a new synthesizer

- to select a moderately large set of waveforms which will be representative (if not individually, then in weighted pairs or triples) of all the different spectra in many tones played at different pitches by different instruments, and
- to analyze each tone so that it can be well approximated by multiple wavetable interpolation using dynamically changing subsets of this set of basis spectra,

the synthesizer, whether implemented in hardware or in software, would have some advantageous characteristics:

- The synthesizer could generate spectra of almost arbitrary complexity without the computational expense of sinusoidal additive synthesis.
- By storing the basis spectra in a bank of wavetables, either in ROM for a hardware implementation or by loading them into memory during the initialization of a software synthesizer, and using digital oscillators which can access multiple wavetables by index [11], the possible delays due to loading wavetables during real-time synthesis are avoided.
- By selecting basis spectra which are generally useful rather than a smaller but more specific set for each tone at various pitches for each instrument, the total amount of memory used to store waveforms would be reduced.

This research focused on finding a way to make such a synthesizer possible. While design decisions were generally predicated on the assumption that the selection of basis spectra and the analysis of tones would be one-time processes, so the fidelity of the results would be more important than computational efficiency, it was also recognized that electronic composers might be experimenting with different sets of tones at different times, so the trade-offs between computation time and accuracy were also explored.

Two goals remained constant: synthesis using the new method should be fast and the control data stream should be sparse. While it was never the goal of this research to find a new audio compression algorithm—such a project would have far more general assumptions about the nature of the audio inputs and would be far more focused on analysis for the purpose of compression—the possibility was considered that a commercial synthesizer might be designed which could load its wavetable bank from an external source (by download or

[15] In fact, this was exactly the case for the set of tones which are represented by the spectrum from the release of a saxophone tone which is stored at index 3 of the wavetable bank for the tones of group 1. (See §4.3.2, §4.3.3, and §5.4).

from a data card) and that users might exchange "patches" (control streams for the synthesis of particular tones) relative to particular wavetable banks; a highly data-reduced control stream would allow real-time streaming of synthesis data over a network, for example.

Risset and Wessel envision an electronic music instrument which would apply an analysis/synthesis method to user-provided sounds. Aware of the computational demands of the analysis stage, they point out that

> analysis can be performed in advance for a corpus of sounds: then one can work live to perform intimate transformations on those sounds. . . . With just a little more engineering innovation, analysis-synthesis technology should accomplish for virtually all sounds the acoustic accuracy of recording playback as used in samplers. Thus research in analysis-synthesis is perhaps now more important than ever. [79, pp. 144, 146]

1.2.2 Indexed Color: An Analogy

Multiple wavetable interpolation operates in a manner analogous to *indexed color* on a video display [12, p. 160ff]. Many applications do not require full access to the millions of colors afforded by the 24-bit color depth of *direct color*, which allocates 8 bits of resolution to each of the red, green, and blue (RGB) values, and would be better served by a method which used less memory (perhaps due to the limited video RAM available on the display card), less disk space to store image files, or less network bandwidth for transmission (as in a Web page graphic). On the other hand, the 256 colors of 8-bit color might not suffice, since the standard set of colors are spread evenly over the spectrum and are unlikely to accord with the colors needed to render the desired image, even if it uses only 256 or fewer different colors. Using indexed color, the 8-bit value associated with each pixel is used as an index into a *palette* (or color lookup table) containing the 24-bit RGB values for the set of colors actually used in the image.

Indexed color is the digital equivalent of a paint-by-numbers kit, in which each area on the canvas to be painted is labelled with a small number, indicating which of the small pots of paint in the kit is to be used to paint that area; each kit includes only those paints needed to color the particular picture featured in that kit.[16]

Although the palette needs to be stored along with the actual image data (unless a standard system palette is used), the datum for each pixel is only one-third the size of that for direct color, so there is typically a net saving of space. If a palette is shared among several images (as it must be in some contexts, such as multiple images on the same Web page), the overall space savings will be greater.

[16] This analogy is due to Chapman and Chapman [12], which source also served as the basis of the rest of this discussion of indexed color.

Rendering an image involves a level of indirection: once the palette has been loaded into the color lookup table, the video hardware must perform a lookup operation which effectively maps the *logical colors* of the image data to the *physical colors* which are driven by the three 8-bit RGB values stored in the color lookup table. It is clear that a logical color value does not identify a color absolutely, but only relative to a given palette.

When converting an image to the indexed color format, if the image includes a color which is not in the palette, it can be approximated in one of two ways:

- The color value may be replaced by the index of the nearest color which is in the color lookup table (where "nearest" may be determined as a simple Euclidean distance or may take into account the non-linearities of our perception of color).

- An unavailable color may be approximated by *dithering*: a group of pixels of the desired color are replaced by a pattern of dots of several different colors which, as a result of optical mixing in the eye of the viewer, are collectively perceived as the desired color (or a close approximation of it).

1.2.3 The New Analysis/Synthesis Method: The Analogy Applied

Optimized multiple wavetable interpolation will be explained in detail in Chapter 3, but can be understood in a preliminary way by comparison with indexed color. Discussion of the analogy is facilitated by the fact that the term "tone color" has long been used as a synonym for "timbre," so "color" may be understood as representing "spectrum" or "waveform" in the following; similarly, "palette" might stand for a wavetable bank and "rendering an image" is analogous to synthesizing a tone. In this context, the following similarities might be highlighted:

- The method selects colors for the palette from the set of those actually used in the images/tones to be rendered. In this regard, the proposed method compares more directly to the selection of a palette to be used in rendering multiple different images, such as the set of images on a Web page, than to the storage of a customized palette in a GIF file, since the focus of this research is on demonstrating that the waveforms of a well-selected wavetable bank can be used to synthesize a wide variety of instrumental tones.

- In the data-reduced representation, colors are referred to indirectly by index; the full specification of each color is stored only once per color, in the table of colors, and a logical color is mapped to its physical counterpart by indexing.

- The rendering of data is relative to a particular palette, but the same palette can be used to render multiple images/tones.

- Colors which are not in the palette must be approximated. In selecting waveforms to be included in a wavetable bank, the proposed analysis/synthesis method uses clustering to group spectra into classes and selects one spectrum from each class to represent all the spectra of that class; this is analogous to choosing the closest color in the palette to approximate a desired color. The method also uses weighted combinations of multiple wavetables from the wavetable bank to approximate spectra, a process which is similar to dithering in the visual media.

1.2.4 Contribution

The primary contribution of this research is a new analysis/synthesis method, optimized multiple wavetable interpolation, which extends the previous work on multiple wavetable interpolation to facilitate the synthesis of tones from a wide variety of different instruments from the same set of wavetables.

The proposed method differs from existing methods primarily due to its goal of resynthesizing a large number of tones from a fixed number of wavetables. Some existing methods (see §2.4.7 and §2.4.3 below) have been tested with ten to fourteen instrument tones, and have resynthesized them with four or five wavetables; Horner [42] used up to ten wavetables in the course of synthesizing single tones by multiple wavetable interpolation, but selected the wavetables from each tone individually. No previous methods have been designed to utilize the similarities between different portions of a large number of different input tones in order to reduce the overall memory requirements for resynthesizing them.

Several components of the new method may be identified as contributions independently of the method as a whole:

- The method of constructing a wavetable bank (§3), including the use of a clustering algorithm (§4.3.1) and the selection of the spectrum nearest the centroid of each class as a class representative (§4.3.3), is a new contribution. Principal components analysis and a genetic algorithm have been used for wavetable selection [41, 44], but no previous study has shown that clustering of spectra according to their normalized harmonic amplitudes could be used effectively for that purpose. Stapleton and Bass [89] (see §2.4.3) grouped tones into classes according to the instruments which produced the tones, but did not cluster spectra individually.

- The grouping of sample tones by pitch (§4.3.2) and using a different wavetable bank for each group, with different numbers of harmonics in the spectra of each bank, so that the highest expected partial frequency will be less than the Nyquist frequency is a new method. Neither of the

other studies which used the same basis spectra to synthesize different tones from the same instrument [41] or tones from different instruments [89] grouped the tones by pitch; the latter study did observe, however, that their "basis functions must be band-limited to prevent aliasing, resulting in loss of information at higher frequencies for any tone" [89, p. 318].

- The breakpoint matching algorithm (§4 and §3.2), including the use of a multi-level search (§3.2.2 and §4.4.1), the overlapping of wavetable sets (§3.2.3), the method of constructing a directed acyclic graph to represent all possible combinations of wavetable-to-oscillator assignments (§4.4.4), and the use of the single-source acyclic weighted shortest path algorithm for oscillator assignment optimization (§3.2.4), is a significant contribution, in that it yields a globally optimal set of oscillator assignments.

A secondary contribution of this research is a new breakpoint-selection algorithm (§4.2.1) which operates by segment merging; the algorithm has characteristics which make it well suited to use on instrumental tones, especially those with vibrato or other pronounced amplitude changes.

After an overview of Fourier analysis and methods of data reduction and a review of several analysis/synthesis methods in Chapter 2, the proposed new analysis/synthesis method will be presented in detail in Chapter 3, followed by a discussion of significant implementation details in Chapter 4. The results of testing the algorithm on a set of 198 tones produced by 16 different instruments, divided into five groups spanning five octaves, will be presented in Chapter 5. Conclusions and suggestions for further research are offered in Chapter 6.

Chapter 2

Music Analysis/Synthesis: An Overview

2.1 Definition of Analysis/Synthesis

Analysis/synthesis [10] (or analysis/resynthesis [80]) is a concept or general framework encompassing a variety of techniques which share a common three-step process:

1. An input waveform is analyzed.
2. The analysis data is modified.
3. An output waveform is synthesized from the modified data.

Analysis/synthesis can be used for several purposes [78, 96]:

- To reproduce the perceptual features of the input sound from a representation based on a sound model.
- To achieve data reduction, as through linear predictive coding or piecewise linear approximation.
- To produce musically interesting variants of a sound, either to change features such as pitch, duration, articulation, or loudness while preserving its timbral identity or to expand timbral resources.
- To gain insight into the perception or multidimensionality of timbre.

The analysis data may be modified in a variety of ways to create musically interesting effects [22, 65, 72, 78, 80]:

Time scaling Stretch or compress a sound in time without changing its pitch.

Pitch shifting Change the pitch of a tone without time scaling.

Timbral interpolation Interpolate over time between two different spectral envelopes.

Spectrum scaling or shifting Multiply the frequency of all partials by a factor or add a constant to all partials, possibly excepting the fundamental.

Cross-synthesis Use the amplitude envelopes of the partials of one sound to scale or replace those of another sound.

Register extrapolation Extrapolate an instrumental timbre beyond the playable register of the instrument.

Transform percussive sounds into fluid textures Delay the onset time of each partial and smooth partial envelopes to sustain a percussive sound while preserving its frequency content.

Variations of recorded sounds Change selected frequency or amplitude envelopes by editing or multiplications by arbitrary functions.

Hybrid timbres Replace some envelopes from one sound with selected envelopes from another sound.

2.2 Fourier Analysis

The definition of the Fourier transform is

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi ift}dt \tag{2.1}$$

where $x(t)$ is a function (assumed to be periodic and infinite) of time t, $X(f)$ is a function of frequency f, e is the natural base of logarithms, and i is the imaginary unit ($\sqrt{-1}$). $X(f)$ may be interpreted as the *spectrum* of $x(t)$. That is, the Fourier transform may be seen as converting a signal in the time domain into a frequency domain representation, effectively breaking the signal into a set of sinusoids of specified amplitudes and phases which, when "added up," would give the original signal.

If, instead of a continuous function $x(t)$, we have a digital signal $x(n)$ consisting of N samples (indexed by sample index n) of a continuous waveform, the *discrete Fourier transform* (DFT) is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-2\pi ikn/N} \tag{2.2}$$

where k is a frequency index [47].

There are two problems with using the Fourier transform to analyze audio signals of the sort that arise in music and speech applications:

- It assumes the input is a periodic signal; that is, it treats the input as one period of an infinitely periodic waveform. If the input signal is non-periodic (like most music and speech audio input, which changes rapidly in time) or if N is not a multiple of the period length, then the DFT coefficients $X(k)$ cannot be interpreted as the amplitude and phase of the waveform harmonics. In the latter case, the spectrum will include broadband noise called *crosstalk* or *spectral splatter* [48].

- The result is given in terms of equally spaced frequency bins. If given N samples of a digital waveform, the Fourier transform will give the results in terms of the integer multiples of a *fundamental frequency of analysis* which is the frequency associated with the duration of N samples. For example, given 1000 samples of a waveform sampled at 10,000 Hz, the discrete Fourier transform will show how to construct that waveform out of the available harmonics of $10,000/1000 = 10$ Hz [65]. However, the human ear processes audio input in approximately one-third-octave bins [99], where each octave represents a doubling in frequency.

Taken together, these characteristics imply that the frequency resolution of the Fourier transform can be increased by analyzing a greater number of samples, but this will decrease the temporal resolution of the analysis.

The broadband noise which results if the sample length is not a multiple of the period length can be reduced by *linear scaling*—scaling the sample sequence by a linear function so that the beginning and ending values are the same—or by *windowing*—scaling the sample sequence by an N-sequence *window* which typically begins and ends at or near zero and rises gradually to a peak between these points.

While windowing reduces crosstalk by smoothing out the discontinuity caused by viewing the input as a periodic sequence of length N, it spreads the energy in a frequency bin into neighbouring bins. This spectrum leakage is due to *sidelobes* in the spectrum of the window.

The discrete Fourier transform is usually implemented using a divide-and-conquer algorithm—the Fast Fourier Transform (FFT)—which runs in time $O(N \log N)$ rather than the $O(N^2)$ time which would result from a direct implementation of the defining formula in equation (2.2).

2.2.1 The Short-Time Fourier Transform (STFT)

While music signals are typically non-periodic over a broad time scale (say, 250 to 500 ms), the properties of the waveform are relatively invariant over intervals of 10–30 ms in duration [84, for speech signals]. Furthermore, based on evidence that the human ear can only register distinct sensations which are separated in time by 10–20 ms, Gabor [34] theorized that a granular or quantum representation could describe any sound. According to this theory, a music signal will be more amenable to Fourier analysis on a segment-by-segment basis.

The short-time Fourier transform analyzes a signal over a time-invariant interval delineated by a window function $w(n)$ of length N_w (i.e., assumed to be zero everywhere except over the interval $0 \ldots N_w - 1$). The time-dependent DFT, which computes the average spectrum content over the N_w samples in the window, is defined as [21]

$$X(n,k) = \sum_{m=-\infty}^{\infty} x(m)w(n-m)e^{-2\pi ikm/N} \tag{2.3}$$

The window is then moved along the signal by a skip factor S (which is 1 if non-overlapping windows are desired or less than 1 for overlapping windows), and the DFT is applied again.

This procedure reveals how the spectrum of $x(n)$ changes over time. Previous studies have shown that the different partials of a tone have different envelopes: they increase in amplitude at different rates (attack or *rise time*), are relatively constant in amplitude for varying durations (*sustain time*), and then decrease in amplitude at different rates (release or *decay time*) [38, 77].

2.3 Data Reduction

There is an important trade-off in computer music synthesis between the amount of control information required by the technique and the amount of control of the process which the method provides. For example, phase vocoder-based additive synthesis uses far more control data than the number of samples being analyzed and resynthesized; however, the control data is in the form of frequency and amplitude parameters for each sinusoidal partial, and thus may be manipulated for performing spectrum modifications such as pitch shifting and time scaling.

2.3.1 Goals of Data Reduction

Research in data reduction techniques may have a variety of goals, including:

- speeding up data reduction processing
- converting control data into editable form
- facilitating real-time synthesis from reduced data (computational complexity)
- reducing the memory requirements of a synthesis technique
- reducing the communication bandwidth required to transmit control data to a synthesizer
- improving the fidelity of synthesis using reduced data

- determining the relevant features of hearing (especially the invariable elements of sound perception [13]).

2.3.2 Effects of Data Reduction

Different data reduction techniques may have different types and degrees of effect on the resulting sound. We may categorize the degree of difference between an original sound and its resynthesis after data reduction according to the subjective judgment of a human listener. Moore [65, page 215] suggests the scale shown in Table 2.1.

Category	Difference Criterion	Example
0	Physically indistinguishable	Identical waveforms
1	Perceptually indistinguishable	Identical percepts
2	Musically indistinguishable	Musically interchangeable
3	Musically acceptable	Musically substitutable
4	Musically different	Musically distinct identities
5	Musically independent	Completely unrelated sounds

Table 2.1: Categories of musical sound differentiability.

For example, Moore suggests that we might classify as *musically indistinguishable* two musical sound events which are as alike as two such events performed on the same instrument by the same performer who is trying to make them as alike as possible. A data reduction technique might be regarded as *musically acceptable* if the result were clearly audible but no more objectionable than the difference between one performer playing an instrument in a French style and another performer playing the same instrument in a German style. A *musically different* result would sound like a different instrument than the original sound.

Two general approaches to data reduction which might result in perceptually or musically indistinguishable results (categories 1 or 2) are:

Receiver coding Basing the properties of the synthesized sound on the properties of the listener's hearing mechanism.

Source coding Reproducing the acoustic properties of the signal to a degree of approximation that preserves objective properties of the sound source such as its time-varying power spectrum [65, page 216].

Receiver Coding

The restricted bandwidth used for telephony is an example of data reduction based on the knowledge that much of the information required for speech intelligibility is typically below 3 kHz [84]. Receiver coding has also been proposed as a method of achieving the reduced bit rate required for digital audio broadcasting [101].

Receiver coding may also be useful for data reduction in music analysis and synthesis. Psychophysical studies [70, 75, 99] indicate that two frequency components which lie within a *critical band* (about 20% of frequency for frequencies greater than 500 Hz.) of one another mask each other (i.e., their amplitudes are not additive). Similarly, low frequency tones tend to mask high frequency ones. There has been some experimentation with the use of critical-band models for categorizing sounds [24] and for analysis/synthesis [98].

Using additive synthesis techniques, one may choose to omit any component with an amplitude smaller than some threshold on the assumption that such components would be masked by other components. McAulay-Quatieri (MQ) analysis (see §2.4.2 below) uses a threshold to approximate psychoacoustic masking [28].

Receiver coding is a complex and relatively unexplored area, and is not used in this research; however, a critical-band-based error measure could be used instead of the Euclidean measures discussed in §4 and §4.2.2.

Source Coding

In his doctoral research on timbre perception [38], John Grey experimented with replacing the time-varying amplitudes of the various components of harmonic tones produced by 16 different instruments with their piecewise linear approximations. Grey asked listeners trained in music to compare the results of resynthesizing the instrument tones using all the data produced by the analysis technique with those resynthesized using line-segment approximations to the amplitude curves and found that the two cases were difficult for the listeners to distinguish. "This finding suggests that the highly complex microstructure in the time-variant amplitude and frequency functions given by the analysis is not absolutely essential to the timbral quality of the tone, and such a drastic data reduction as was performed in this case will do little harm" [38, p. 40]. Grey also found that eliminating the frequency variations which occurred in various components over the duration of the sound could *not* be eliminated without seriously affecting the discriminability of the tones [40]. J.-C. Risset had previously used the same technique successfully with trumpet tones [77].

John Strawn [90] proposed the use of a syntactic parser to identify breakpoints for line-segment approximation which correspond to features which are defined to be of interest by specifying a grammar for the parser.

Gérard Charbonneau [13] tested three types of more drastic data reduction: using a single amplitude mean curve for all partials, using a single frequency reference curve, and approximating the starting and ending times of each partial with a polynomial. The timing simplification gave the best results, while the amplitude simplification was the most audible.

James Beauchamp [3] found one-to-one relationships between the instantaneous amplitudes of the partials of cornet tones which, given the amplitude envelope of the first harmonic, can be used to represent (in data-reduced

form) and resynthesize realistic-sounding tones. These *nonlinear interharmonic relationships* were well approximated by a third-degree polynomial. Beauchamp [4] also found a relationship between *brightness*—the spectral center of gravity—and RMS amplitude for the cornet and saxophone which provides data reduction and low-error resynthesis with a nonlinear synthesis method.

McAdams et al. [59] tested listeners' ability to discriminate between reference intrumental tones and tones synthesized from simplified analysis data; six basic simplifications were tested, both singly and in combination. Most simplifications were discriminable, and different simplifications were indiscriminable on different instruments. In general, the smoothing of amplitude envelopes after the attack portion, the smoothing of frequency envelopes, and the coherence of frequency envelopes (i.e., using a single frequency envelope for all harmonics) were found to be the least discriminable (in that order).

This research uses source coding of two forms, both of which are due to the use of common breakpoints (i.e., breakpoints which are selected to occur for all harmonics simultaneously) to simplify harmonic amplitude and frequency envelopes:

- Spectral interpolation between breakpoints results in a piecewise linear approximation of harmonic amplitudes. (See §2.4.5 below.) This is essentially the same form of data reduction used by Grey, Risset, and Strawn, except for the requirement of common breakpoints across all harmonics; the increased inaccuracy of the approximation which might have resulted from the use of common breakpoints is avoided by the use of more breakpoints per tone, as discussed in §4.2.3.

- In addition to determining the amplitude of each sinusoidal component of a signal, Fourier analysis yields phase information from which the instantaneous frequency of each partial can be calculated. The difference between the calculated frequency of a partial and its theoretical harmonic frequency (which would be an integer multiple of the fundamental frequency) is called the *frequency differential* of that partial. Since wavetable synthesis precludes the possibility of partials varying in frequency independently, only a single frequency differential can be used to control the frequency of the synthesized signal at a given point in time. Typically, an average of the differential frequencies of all the harmonics is used, weighted according to the amplitude of each harmonic.

  The use of a single weighted-average frequency differential at each breakpoint (see §2) is similar to Strawn's frequency simplification and essentially equivalent to the use of coherent frequency envelopes as tested by McAdams et al. This simplification eliminates the possibility of inharmonicity between partials, which is a salient perceptual feature for some instrumental tones, especially for piano tones [9, 51, 86], but not for others [31].

2.4 Methods of Analysis/Synthesis

A variety of analysis/synthesis systems have been proposed, the most significant of which are reviewed here.[1]

2.4.1 Phase Vocoder Analysis/Synthesis

If the output of a short-time Fourier transform (STFT) analysis is converted from rectangular to polar coordinates—the *magnitude* (radial position) and *phase* (angular position) of each $X(k)$—the time-varying amplitude and frequency parameters needed to drive an oscillator for resynthesis of the original signal may be calculated [65]. Thus, STFT analysis may be viewed as modeling the input signal in terms of multiple parallel channels, each analyzed by a bandpass filter [21, 37].

Since the origin of this method was for reducing the bandwidth required for telephony, the technique was called a *voice encoder* [23]. The *phase vocoder* [21, 29, 37, 61, 73] is a form of voice encoder which preserves phase information, thus allowing the perfect resynthesis of the original signal (provided that the window function is zero at integer multiples of N).

The amplitude and frequency parameters may also be modified prior to using them as control signals for a bank of oscillators in resynthesis. For example, the pitch of an input signal may be modified without changing its duration (*pitch shifting*), and vice versa (*time scaling*) [21].

The intermediate data may also be encoded (e.g., by piecewise linear approximation) to reduce the storage needed for a parameterized representation of a musical sound [65].

It is also possible to resynthesize the input signal to a phase vocoder without converting the STFT output to amplitude and frequency envelopes by *overlap-add resynthesis*. This method uses the inverse Fourier transform to reconstruct each frame of the windowed signal, then overlapping and adding these resynthesized windows. This method is computationally less expensive than oscillator-bank resynthesis, but modifications to the output of the analysis phase typically create audible side-effects on resynthesis.

The *heterodyne filter* and *channel vocoder* are earlier techniques related to the phase vocoder.

2.4.2 Peak-Tracking Phase Vocoder

McAulay and Quatieri [60] developed an analysis method (referred to as "MQ analysis" or *sinusoidal modeling*) based on the STFT which addresses the problem of spectral splatter due to the sidelobes of the window function. In

[1] See also the review by Risset and Wessel [79, §XIV] of the use of analysis and synthesis in perceptual studies of timbre and the report [96] on the initial phase of a comparison of six analysis/synthesis systems using common input sounds and a common output format. Jaffe proposed a set of criteria for evaluating synthesis techniques [49].

MQ analysis, the peaks in the output from STFT analysis are located and tracked from frame to frame by a matching procedure. Amplitude, frequency, and phase envelopes are calculated by interpolation for these tracks, and can be used to drive sinusoidal oscillators for resynthesis. Spectral components other than the peaks in the spectrum are discarded, since they are assumed to be noise.

Since this method explicitly tracks the changes in frequency of the most prominent peaks in the spectrum over time, analysis/synthesis systems using this algorithm are called *tracking phase vocoders.* The analysis of those sinusoidal components which are tracked is more accurate than phase vocoder analysis [80], and is well suited to the representation of inharmonic and quasi-harmonic partials (as are found in piano tones [9]) and pitch fluctuations (such as vibrato) [35, 57].

However, the MQ analysis technique is not as robust as phase vocoder analysis, not only when applied to harmonic tones with little vibrato [5], but also for percussive sounds [6]. The primary problems are (a) deciding which amplitude peaks represent sinusoids and which are noise, especially in low-amplitude portions of tones, and (b) peak tracking, especially when one track appears to die, only to be reborn at about the same frequency a few frames later. As implemented by Maher and Beauchamp in the program `mqan` of the *SNDAN* sound analysis/synthesis suite [5, 56], peak-picking uses a relative threshold defined in terms of the magnitude of the largest peak in the frame; Fitz, Walker and Haken [28] improved on this scheme in their implementation of *Lemur* by applying a different relative magnitude threshold in each of a number of logarithmically sized bins. Serra and Smith's *PARSHL* system [88], which uses a method very like MQ analysis, allows tracks to lie dormant for several frames before dying out, but synthesizes the dormant segments at zero amplitude; *Lemur* reduces the audible effects of frequent track deaths and rebirths by using two different thresholds, a higher one for track births and a lower one for track deaths [28]. The program `hmm` from the Institut de Recherche et Coordination Acoustique/Musique (IRCAM) takes a statistical approach to peak tracking, using hidden Markov models to find globally optimal tracks [18].

2.4.3 Karhunen-Loève Synthesis

Stapleton and Bass [89] proposed a synthesis technique which uses a type of principal components analysis, the Karhunen-Loève transform, to find a small set of basis functions from which a set of harmonic input tones may be resynthesized by multiple wavetable synthesis. This analysis method operates in the time domain, not in the frequency domain as do Fourier-based methods. It requires that segments of each input tone be normalized to have the same fundamental period, be band-limited by low-pass filtering, and phase aligned.

This method was applied to a set of fourteen representative instrumental tones. It was found that the synthesis was improved by grouping the tones

into three classes with a correlation algorithm prior to determining the basis functions. The classes were:

1. flute, guitar, marimba, violin, plucked violin, diapason (organ pipe);

2. trumpet, French horn, trombone; and

3. clarinet, oboe, bassoon, tenor saxophone, alto saxophone.

Each instrument required two to four basis functions for high-quality synthesis. It was found that basis functions which work well for a given tone will not necessarily work well for audibly similar tones.

The signal matching required by this method is computationally expensive. Furthermore, modifying the amplitude envelopes during resynthesis can result in nonlinear changes in harmonic amplitudes due to phase cancellation [41].

2.4.4 Spectral Modeling Synthesis

Spectral modeling synthesis (SMS) models time-varying spectra as a collection of sinusoids (the deterministic component) and a time-varying filtered noise component (the stochastic part) [87]. Since noise consists, in principle, of sinusoids at every frequency, simulating the noise component of a sound by additive synthesis is very expensive. SMS tracks the peaks in a short-time Fourier transform analysis, removes them by spectral subtraction, then models the residual (the "noise floor") as white noise passed through a time-varying filter.

SMS captures the perceptual characteristics of a wide variety of sounds, and the analysis method yields intuitive parameters. However, it represents a sound by two different components which, if subjected to different transformations, may not fuse into a single entity on resynthesis.

An extension of SMS has recently been proposed which would add a third component to the sound model: transient modeling synthesis (TMS) [92]. TMS models transients—short-lived signals—using sinusoidal modeling in the discrete cosine transform (DCT) domain, since impulses in the time domain correspond to slowly varying sinusoids in the DCT domain.

2.4.5 Spectral Interpolation Synthesis

Serra, Rubine and Dannenberg [86] proposed a method of generating time-varying sounds by interpolating between analyzed spectra. They begin by analyzing a signal using a pitch-synchronous short-time Fourier transform with a low time sampling rate (one Fourier analysis per period of the fundamental frequency). This gives a harmonic spectrum—a vector of amplitudes, one amplitude per harmonic—for each period of the tone; the list of spectra with their time indices is called the spectral envelope of the tone.

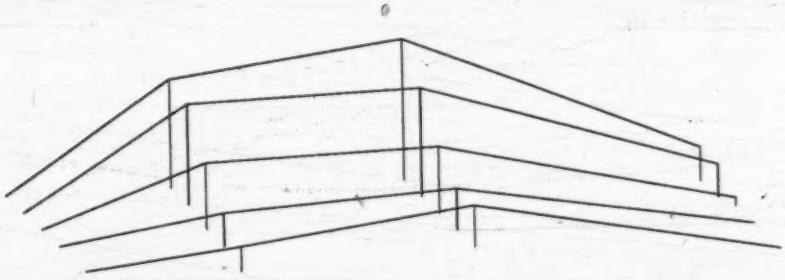


Figure 2.1: Three spectral ramps. (Based on a figure in [86].)

The spectral envelope is then data-reduced using line-segment approximation. However, the breakpoints that define the piecewise linear function for each harmonic are simultaneous, creating *spectral ramps* as illustrated in Figure 2.1. Breakpoints are selected using an error threshold: if the sum of the errors on the individual spectra within a spectral ramp is less than the tolerated threshold, the spectral ramp is extended to the next period.

The waveform is then resynthesized using a waveform interpolation oscillator consisting of two wavetables L and R, each with an amplitude scale factor (c and d respectively), and with a common index function. To begin, the left wavetable L is loaded with a waveform corresponding to the first spectrum of the analyzed signal (created by additive synthesis, but ignoring the initial phase differences between different harmonics and the phase shifts of each given harmonic in time), and its amplitude scale factor c is set to one. The right wavetable R is loaded with the waveform of the spectrum at the first breakpoint, and its amplitude scale factor d is set to zero. Output samples are then generated by extracting samples from the two wavetables, scaling each sample by its associated amplitude scale factor, and adding them together. As the output sample index progresses from zero to the index of the first breakpoint, the amplitude scale factors are linearly interpolated such that c will reach zero and d will reach one by the first breakpoint. At the breakpoint, the waveform in table L is replaced by the waveform of the spectrum at breakpoint two, and the adjustment of the amplitude scale factors reverses direction so that c returns to one and d returns to zero by the second breakpoint. Thus, spectra between the breakpoints are linear interpolations of the spectra at the previous and next breakpoints.

Serra et al. also experimented with data reduction using nonlinear spectral interpolation, in which the amplitude scaling functions $c(n)$ and $d(n)$ are arbitrary functions of time, rather than being constrained to be opposite linear ramps summing to unity. In this case, breakpoints are separated by arbitrarily large time intervals and the square error within the spectral path is minimized by choosing the best coefficients $c(n)$ and $d(n)$. If the error is larger than a specified threshold, a smaller interpolation duration is tried. Special techniques must be used to avoid generating clicks when switching waveforms in the oscillator wavetables.

Horner, Cheung, and Beauchamp [45] used both a greedy algorithm and a genetic algorithm (GA) to select the N best interior breakpoints common to all harmonics of a sound such that line segments between these breakpoints would give the best piecewise linear approximation (PLA) to the amplitude envelopes of all harmonics. By using common breakpoints instead of finding the best PLA for each harmonic individually, the tone can be resynthesized using spectral interpolation synthesis instead of additive synthesis. The genetic algorithm gave the best results when fewer than ten breakpoints were specified; the greedy algorithm performed as well as the GA for ten or more breakpoints, with less computation.

2.4.6 Analysis-by-Synthesis/Overlap-Add

George and Smith [35] proposed an analysis/synthesis method which allows the use of overlap-add resynthesis without the audible artifacts that usually result from the modification of the analysis data. Their system—analysis-by-synthesis/overlap-add (ABS/OLA)—uses an iterative algorithm which finds the best least-squares approximation of a single sinusoid to the input signal, subtracts the resynthesized sinusoid from the original signal, then finds the best fit of another sinusoid to the residual, repeating this process until the desired accuracy is attained.

Because analysis-by-synthesis removes the crosstalk associated with each component as it is estimated, the ABS/OLA system provides an accurate representation of inharmonic and quasi-harmonic tones and tones which vary in pitch. It allows time- and frequency-scale modifications without the reverberant artifacts associated with modified phase vocoder resynthesis. Synthesis is relatively efficient, since it uses the inverse FFT, but the analysis procedure is relatively inefficient.

2.4.7 Multiple Wavetable Synthesis

Multiple wavetable synthesis is an additive synthesis technique based on the sum of fixed waveforms or periodic basis functions with time-varying weights. Unlike classical additive synthesis in which the waveforms to be added are simple sinusoids, multiple wavetable synthesis loads each wavetable with one cycle of a waveform of arbitrary complexity. Typically, the waveforms to be added are themselves the fixed weighted sum of several harmonic sine waves; the spectrum produced by a particular set of harmonic weights is referred to as a *basis spectrum*. If the sets of harmonics for the various waveforms are disjoint, the method is termed *group additive synthesis* [52].

The principal advantage of multiple wavetable synthesis is its efficiency, since the number of wavetables used is typically much smaller than the number of sine waves that would be used in classical additive synthesis. The principal difficulty is that, for an arbitrary small set of wavetables, most time-varying spectra cannot be approximated very closely by a linear combination of these

wavetables, even if their weights are time-varying. Thus the basis spectra must be chosen carefully and their weights appropriately manipulated when synthesizing dynamic spectra.

Andrew Horner [41, 44] compared two methods of efficiently determining the basis spectra and their time-varying amplitude scale factors to best match the original time-variant spectrum of a signal:

Genetic-algorithm-based selection A genetic algorithm (GA) was used to select basis spectra from the set of discrete-time spectra found by short-time Fourier analysis of the original sound (which Horner calls "spectral snapshots"). The GA attempts to find a set of basis spectra which work well over the the whole duration of the tone, according to a fitness function (a relative error measure, to be discussed in §4.2.2) which measures the quality of the match between a candidate synthetic signal and the original signal.[2]

Principal-components-based matching Basis spectra were determined by a statistical factor analysis procedure, *principal components analysis* (PCA), which derives basis spectra which are optimal in the sense that they capture the maximum variance of the analyzed tone. The basis spectra found are also guaranteed to be orthogonal to one another. These spectra may have a rather artificial relation to the original tone, since generally no basis spectrum will resemble any of the actual analysis spectra; in fact, basis spectra typically include negative amplitudes for some sinusoidal components.

To minimize the cost of repeatedly evaluating the fitness function in the GA-based method, the match between the weighted basis spectra and the spectra of the original tone was not evaluated at every analysis frame, but at twenty selected match points: ten equally spaced in the attack portion and ten equally spaced across the rest of the tone.

Both approaches resulted in perceptually similar matches when four or more basis spectra were used. It was found that four or five basis spectra were adequate for achieving excellent simulation. When using fewer basis spectra, the GA-based approach gave better results. Although PCA minimizes the time-averaged mean-square error between the original and the matched spectra, since low-amplitude spectra in the attack and decay portions of a tone are perceptually more important than their higher amplitude steady-state counterparts, PCA tends to result in relatively large errors during these critical portions of the tone.

Horner also experimented with extending these spectral matching techniques to multiple non-simultaneous tones from the same instrument. He

[2]The relative error measure actually used calculates the error at only a limited number of representative spectra from the sound being matched, rather than using all of the analysis frames, in order to reduce the cost of computing the measure.

found that five basis spectra were sufficient to provide a good match to ten English horn tones spaced a minor third apart over more than a two-octave range; six spectra sufficed for twelve similarly spaced trombone tones. For the violin, it was better to subdivide the range of fourteen tones into two sets of seven tones; five-table matches then sufficed for each subrange.

2.4.8 Bandwidth-Enhanced Additive Modeling

Fitz and Haken [25, 26] extended the MQ analysis used in *Lemur* (previously discussed in §2.4.2) to allow the modeling of noise (non-sinusoidal components) in a unified model (in contrast to the two-part model of SMS [§2.4.4]) by associating a bandwidth with each track. After sinusoidal components have been extracted by the peak-tracking of MQ analysis, the bandwidth-association algorithm distributes the remaining spectral energy in a frequency region among the sinusoids as bandwidth. A bandwidth-enhanced sinusoidal oscillator performs spectral line widening [78] on resynthesis by stochastic modulation [26].

More recently, Fitz et al. [27] introduced the *reassigned bandwidth-enhanced additive model* in a publicly available[3] C++ class library called *Loris*. This model uses the time-frequency reassignment method of Kodera et al. [53] to improve the time resolution of transient events. Partial tracks are also broken at the time of a transient in order to preserve phase accuracy without using cubic interpolation, as is done in MQ analysis; this use of synchronized breakpoint envelopes allows the preservation of transients under transformation.

Although it provides a homogeneous sound model which "can represent a great variety of sounds at high fidelity without sacrificing the intuitive sense of the sinusoidal model" [26, p. 385], bandwidth-enhanced sinusoidal synthesis suffers from the same problem as other sinusoidal additive synthesis techniques: it is "computationally very expensive" [25, p. 154].

2.4.9 Multiple Wavetable Interpolation

Multiple wavetable interpolation is a form of analysis/synthesis in which the recorded sound is reduced to a set of breakpoints by piecewise linear approximation of the spectral envelopes of its harmonics, the spectrum at each breakpoint is matched by determining weightings for a small number of wavetables, and the sound is resynthesized using multiple wavetable additive synthesis by interpolating between the weightings for each wavetable at consecutive breakpoints.

In contrast to multiple wavetable synthesis (§2.4.7), which uses all of the selected basis spectra in each match, multiple wavetable interpolation uses only a subset of the basis spectra at each match point (breakpoint). As will be discussed more fully in item 4 of §3.1, if the subset of the wavetables used at one breakpoint differs from the subset used at the next breakpoint, two or

[3] Available online at `http://www.cerlsoundgroup.org/Loris` (2002-10-01).

more oscillators must be used to crossfade the changing wavetables between match points in order to avoid audible clicks and spectral discontinuities.

Horner [42] tested multiple wavetable interpolation on three different mid-range instrumental tones: a trumpet F4, a piano C4, and a muted trombone B♭3. Basis spectra were selected using the genetic-algorithm-based method discussed in §2.4.7; only the first 20 harmonics of each spectrum were used to evaluate the quality of the matches to the spectra at selected match points (10 equally spaced in time in the attack portion and 10 similarly spaced through the rest of the tone).

Horner dealt with the need to crossfade between changing wavetables by imposing two simple restrictions: if synthesis is to be performed with N oscillators, only $N-1$ wavetables may be used at each match point, and only one wavetable may be changed from one match point to another. His method starts by selecting by enumeration the best combination of $N-1$ wavetables to use at the breakpoint with the peak RMS amplitude, then works backward and forward to neighboring match points, changing at most one of the wavetables, using enumeration to decide what change to make, if any.

Horner tested this method using various combinations of between two and seven oscillators[4] and two to ten wavetables. In comparison to simple (two-oscillator) wavetable interpolation, the use of three or four oscillators provided significantly reduced error; using more than six tables with four oscillators provided only modest error reductions. Compared to multiple wavetable synthesis (or "wavetable matching"), multiple wavetable interpolation provides a way to reduce the computational cost of synthesis by using extra storage space: the use of an additional wavetable with one fewer oscillators gave about the same degree of accuracy. In listening tests, a four-oscillator, five-table match to the trumpet tone was found to be indistinguishable from the original tone, where a match was deemed to be indistinguishable if the average listener confused the synthesized tone with the original tone at least half the time; all three- or four-oscillator matches of the muted trombone tone were found to be indistinguishable, but all synthetic piano tones were correctly identified in at least 70% of trials.

Hybrid Sampling/Wavetable Synthesis

Yuen and Horner [97] propose a hybrid synthesis model which uses *sampling* (see §1.1.3, note 14) for the attack segment and multiple wavetable interpolation for the sustain and release segments, crossfading between the samples and the wavetables in a short transition section. The problem with splicing a sampled attack on a synthesized sustain is that, due to slight frequency fluctuations in the attack segment, each partial may have a different phase and frequency at the splice point; a method is proposed to find the phase values which will cause the least phase cancellation in the transition segment.

[4] The two-oscillator case is similar to spectral interpolation synthesis (see §2.4.5) except that basis spectra are selected differently and may be reused at subsequent match points.

Chapter 3

Optimized Multiple Wavetable Interpolation

The goal of optimized multiple wavetable interpolation analysis/synthesis is to select a number of *basis spectra* and to analyze a number of instrumental (or other quasi-harmonic) tones relative to the basis spectra such that the tones can be resynthesized by interpolating between consecutive pairs of a sequence of spectra, each spectrum of which is the weighted additive combination of multiple (typically, two to five) wavetables from the wavetable bank.

3.1 Overview of the Process

Optimized multiple wavetable interpolation analysis/synthesis is a multi-stage process, as illustrated in Figure 3.1 and described below.

1. **Analysis:** Each tone (waveform) is analyzed using a short-time Fourier transform (typically either a period-synchronous phase vocoder or extended MQ analysis). This analysis groups the samples of the waveform into overlapping frames, the size of which is determined by the expected fundamental frequency of the tone, and produces an amplitude and a differential frequency for each harmonic in each frame. The number of harmonics for which data are produced is determined by the Nyquist frequency (half the sampling rate). Figures 3.2 and 3.3 show time-varying spectrum plots of the first 20 harmonics of period-synchronous phase vocoder analyses of a French horn and a saxophone, respectively, playing the pitch G3.

2. **Breakpoint selection:** The volume of data is reduced by selecting certain frames as the *breakpoints* of a piecewise linear approximation (PLA) of the harmonic amplitude envelopes. Wavetable interpolation requires the use of shared breakpoints—each breakpoint is common to all harmonics—which also gives greater data reduction than using independent breakpoints for each harmonic, since only one set of breakpoint

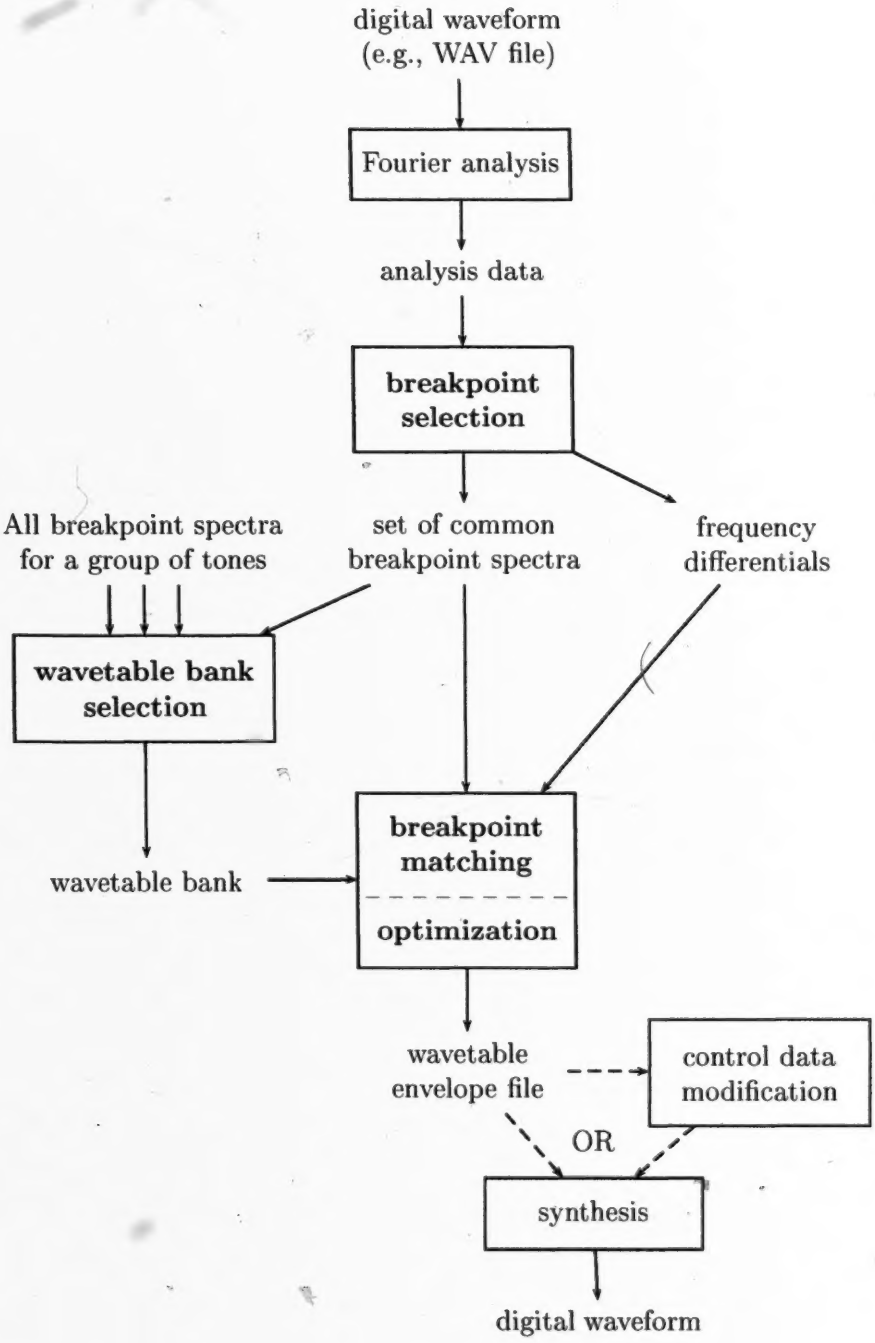


Figure 3.1: The stages of optimized multiple wavetable interpolation analysis/synthesis. New methods are proposed for the stages which are highlighted.

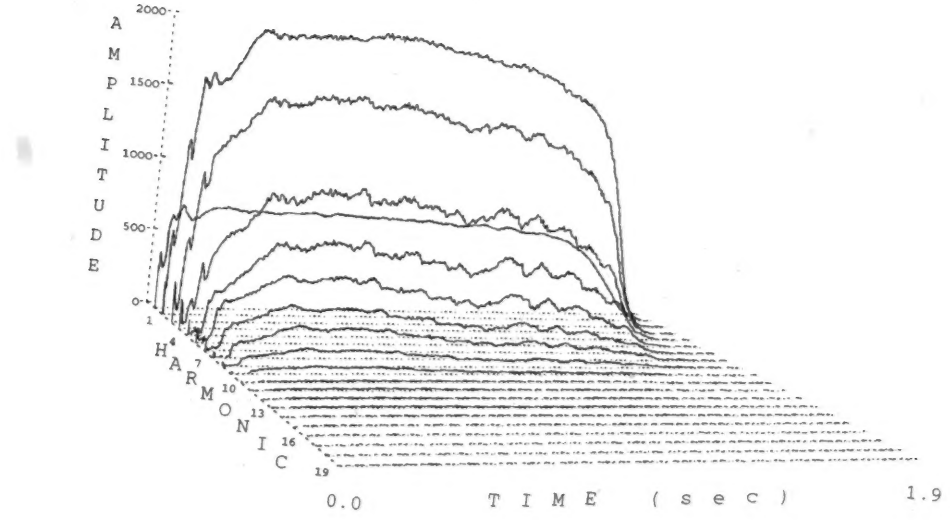


Figure 3.2: Phase vocoder analysis of a French horn playing pitch G3.

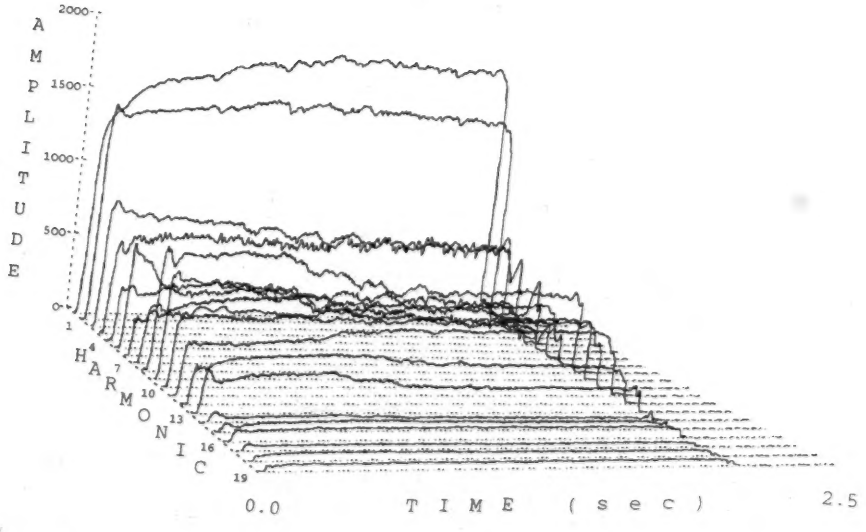


Figure 3.3: Phase vocoder analysis of a saxophone playing pitch G3.

times must be stored [43]. Serra, Rubine, and Dannenberg [86] refer to the segments of a PLA with simultaneous breakpoints as *spectral ramps*.

For this research, a new breakpoint-selection algorithm was developed which is very efficient computationally and which selects shared breakpoints such that the extremes—the peaks and valleys—of harmonic amplitude envelopes are well-modeled by the PLA's generated. The characteristics and implementation of this algorithm, which operates by segment merging, are discussed in §4.2.

Figures 3.4 and 3.5 show spectrum plots of the first 20 harmonics of 24-breakpoint piecewise linear approximations of the time-varying spectra of the French horn and saxophone tones shown in Figures 3.2 and 3.3.

A single weighted-average frequency differential is also computed and stored for each breakpoint, since wavetable interpolation requires that the corresponding harmonics of the wavetables involved in the interpolation be in phase [86].

Unless otherwise specified, references to breakpoints and to numbers of breakpoints are to *internal* breakpoints. The first and last segments of the PLA begin and end, respectively, at implicit *external* breakpoints at time 0 and at a final time index corresponding to the duration of the tone. The number of segments in a PLA is thus one greater than the number of (internal) breakpoints.

3. **Wavetable bank selection:** A number of spectra are selected to comprise a set of basis spectra which will be used in weighted additive combinations to approximate the actual spectra at each breakpoint of each tone to be synthesized. Typically, these basis spectra are selected from the breakpoint spectra chosen by PLA in step 2, but they could be selected by other means, including spectral principal components analysis (PCA) [83], a genetic algorithm [41, 44], or by hand-selection [86, §2.2]. For this research, the basis spectra were selected by using a clustering algorithm on the breakpoint spectra and choosing a representative spectrum from each cluster. The set of basis spectra was then augmented with some hand-picked spectra which were selected in order to reduce the approximation error for certain waveforms.

   The basis spectra are here collectively referred to as a *wavetable bank*, although for the purposes of breakpoint matching (step 4) they are initially represented in the frequency domain as vectors of harmonic amplitudes. At synthesis time (step 6), each vector is converted to an actual wavetable—a table of the time-domain amplitude values of one cycle of the waveform—for use by a table-lookup oscillator (as discussed in §1.1.3).

   If the tones to be synthesized have fundamental frequencies which span a broad range of frequencies, it will be necessary to partition the tones

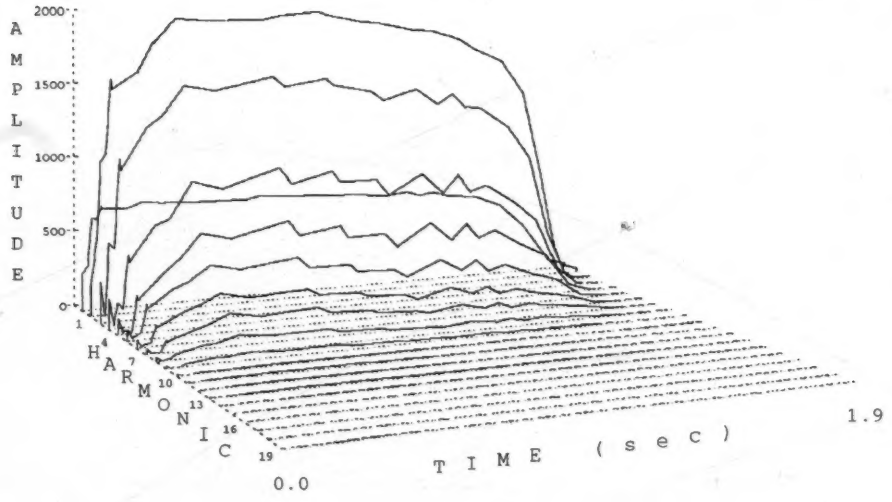


Figure 3.4: Piecewise linear approximation of a French horn playing pitch G3.

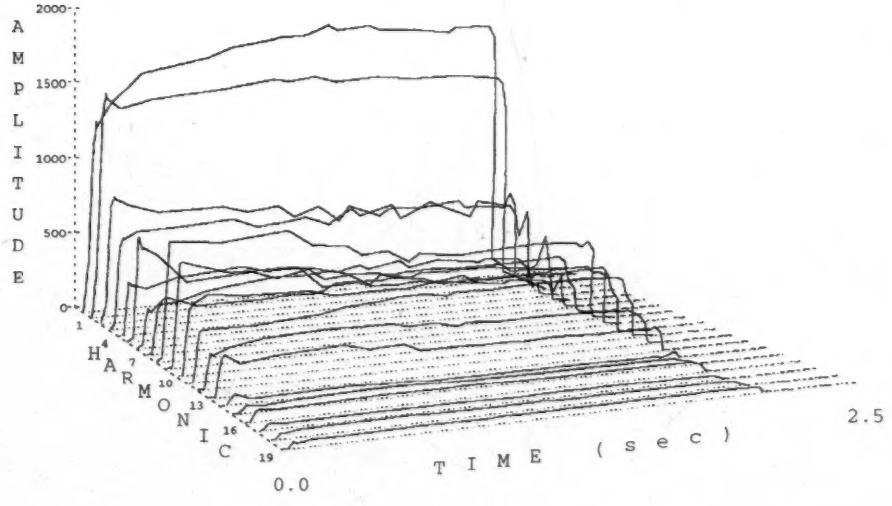


Figure 3.5: Piecewise linear approximation of a saxophone playing pitch G3.

into groups which span a smaller range of fundamental frequencies and to select a different bank of wavetables for each group, due to the restrictions imposed by the sampling theorem. If a spectrum were selected from a low-pitched tone for inclusion in the wavetable bank, complete with all its harmonics, and was then used in synthesizing a tone at a higher pitch (i.e., with a higher fundamental frequency), the upper partials would wrap around the Nyquist frequency, creating synthesis artefacts. If all the spectra in the wavetable bank were band-limited to the frequency range between the highest expected fundamental frequency and the Nyquist frequency, then all the energy in the upper harmonics of the lower-frequency tones would be lost on resynthesis, resulting in audibly degraded tone quality. For this research, the sample tones, spanning five octaves, were partitioned into five groups so that more partials could be retained in the wavetable banks for the lower-pitched tones than for the higher-pitched ones.

4. **Breakpoint matching:** Given a particular wavetable bank, a set of breakpoint data representing a particular tone, and the number of oscillators (N) to be used in resynthesis, the breakpoint-matching algorithm selects, by index, at most N wavetables from the bank which, in weighted combination, best match the spectrum at each breakpoint according to some error measure. Ideally, the error measure used to select breakpoints and to evaluate matches to breakpoint spectra would be based on a psychophysical model which takes into account masking, loudness, and critical bands [100]. Lacking a verified model, a Euclidean measure is used for both, as discussed in §4.2.2. This is consistent with Plomp's finding that Euclidean spectral differences are strongly correlated with a Euclidean timbre space derived by multidimensional scaling from a timbre dissimilarity listening test using complex musical tones [71].

   A different set of wavetables can be selected for use at each breakpoint, subject to the restriction that one cannot suddenly switch from using one wavetable at one breakpoint to a different wavetable at the the next breakpoint, since audible clicks and spectral discontinuities would result [42]. Instead, if a wavetable is used with a non-zero weight at one breakpoint and not used at the next, it must be faded out to zero by the next breakpoint; similarly, if a new wavetable is selected for use at a particular breakpoint, it must be faded in, beginning with a zero weight at the previous breakpoint. It is for this reason that it was specified that the matching algorithm selects *at most* N wavetables for use at each breakpoint, given N oscillators: if the set of wavetables to be used at one breakpoint differs from the set to be used at the subsequent breakpoint, then fewer than N wavetables must be selected for either or both of the breakpoints.

   - If N wavetables are selected for breakpoint B_i, then the same set

of N wavetables or any subset of those wavetables (and no others) must be selected for breakpoint B_{i+1}, since all N oscillators are in use at B_i and, to avoid perceptible discontinuities, each must continue to be used either to interpolate between two weightings of the same wavetable or to fade its assigned wavetable to a weighting of zero, as illustrated in Figure 3.6, parts (a) and (b). (The dotted line from a closed circle to an open circle, with a downward-pointing arrow from the closed circle, in part (b) of the figure is intended to suggest the fade-out of a wavetable to zero amplitude.) Conversely, if M wavetables, $M < N$, are selected for breakpoint B_i and N wavetables are selected for breakpoint B_{i+1}, then the set of wavetables at B_{i+1} must be a superset of those at B_i, since $N - M$ oscillators must be used to fade in the new wavetables selected for B_{i+1}, as illustrated in Figure 3.6(c). (The arrows in this diagram are intended to suggest the fade-in of wavetables by oscillators which were at zero amplitude at the previous breakpoint.)

- If $N-1$ wavetables are selected for breakpoint B_i and $N-1$ wavetables are selected for breakpoint B_{i+1} such that $N-2$ of the wavetables are the same at both breakpoints, then all N oscillators will be in use for the segment between B_i and B_{i+1}, since $N-2$ oscillators will be used for the wavetables in common, one will be used to fade out the wavetable which is not in the set of wavetables selected for breakpoint B_{i+1}, and the last oscillator will be used to fade in the new wavetable selected for B_{i+1}, as illustrated in Figure 3.6(d).
- If $N-1$ wavetables are selected for use at breakpoint B_i and $N-2$ wavetables are selected for breakpoint B_{i+1} such that only $N-3$ of the wavetables are used at both B_i and B_{i+1}, then two oscillators must be used to fade out the expiring wavetables and one must fade in the new wavetable (or vice versa), as illustrated in Figure 3.6(e).
- If N is even, then it is possible that $N/2$ wavetables may be selected for breakpoint B_i and a completely different set of $N/2$ wavetables selected for breakpoint B_{i+1}, in which case all N oscillators will be used to crossfade from the first set of wavetables to the second, as illustrated in Figure 3.6(f).

As discussed in §2.4.9, Horner [42] proposed a method for deciding which subset of the basis spectra to use at each breakpoint (or "match point") in the tone, but his method is locally optimal at only one breakpoint (the point of the peak RMS amplitude) and is restricted to selecting $N-1$ wavetables at each breakpoint such that $N-2$ of them are the same at neighboring breakpoints.

The new method proposed in this thesis is not subject to the restriction of selecting $N-1$ wavetables at each breakpoint and finds a globally optimal set of matches across all breakpoints, given a particular error

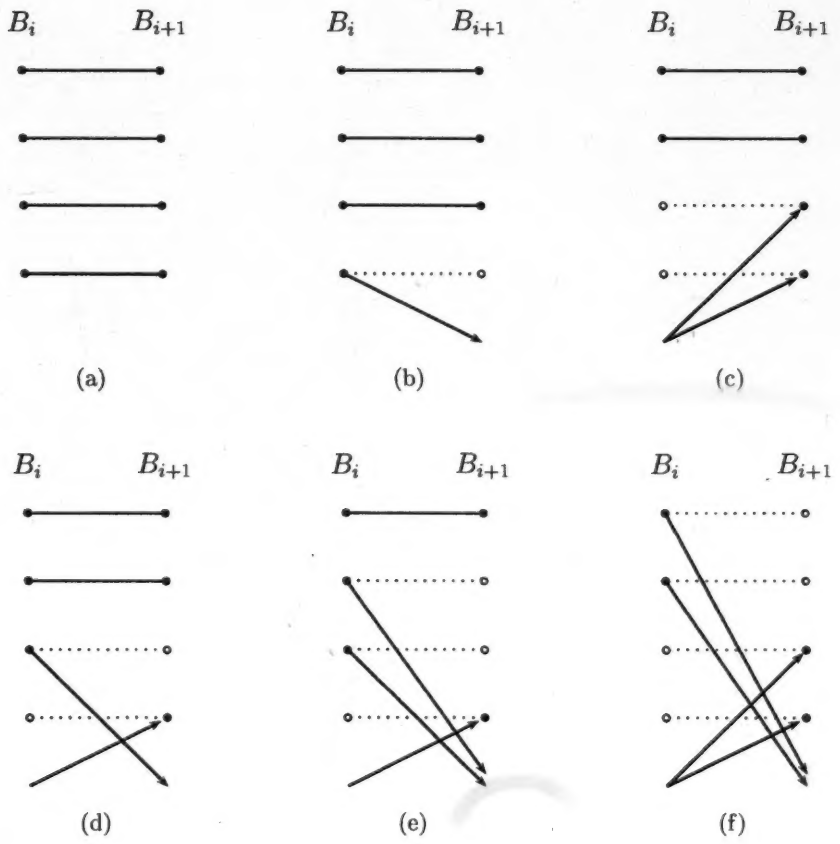


Figure 3.6: Possible oscillator assignments with four oscillators.

measure and a specified method of choosing an initial best match of a specified size at each breakpoint. This method will be presented in more detail in §3.2 below.

The output from this stage of the algorithm is a sequence of records, one per breakpoint, where each record consists of the time index of the breakpoint which it represents, the weighted average frequency differential at this breakpoint, and a set of tuples, one per oscillator, each consisting of a wavetable number and the weighting (amplitude factor) for that wavetable. The start of a fade-in is represented by the number of the wavetable to be faded in and a weighting of zero. Wavetables are numbered from one to the size of the wavetable bank; a wavetable number of zero indicates the end of a fade-out which is not immediately followed by the fade-in of another wavetable.

5. **Modification of analysis data** (optional): If optimized multiple wavetable interpolation is being used as a technique in a computer music composer's toolkit, it is at this point that the analysis data would be modified to create a new sound by synthesis. One can imagine a graphical tool which would read a wavetable bank file and a file of analysis data (which we might call a *wavetable envelope file*, since the weightings specify an amplitude envelope for each wavetable used in breakpoint matching) and display the tone which is specified by the combination of the two in a time-varying spectrum plot such as the one shown in Figures 3.2 and 3.3; the composer might then be able to select different wavetables from the wavetable bank to be used at one or more breakpoints or change the weightings of some tables and see the result by a dynamic update of the spectrum plot.

   For the purposes of the present research, a simpler expedient was chosen: a utility reads a wavetable bank file and a wavetable envelope file and writes a *Csound*[1] unified file (which combines both the *score file* and the *orchestra file* of earlier versions of the *Csound* digital sound synthesis program). This text file can then easily be edited to experiment with different types of modifications.

6. **Synthesis:** The data-reduced and possibly modified analysis data can then be used as a control stream for a custom synthesizer, implemented either in hardware or in software, or, as was done in this research, be converted by the utility mentioned above to a *Csound* unified file and synthesized using *Csound*. The synthesis model is illustrated in Figure 3.7. The figure shows a three-oscillator synthesizer; the model may be reduced to two oscillators or extended with additional oscillators in

[1] *Csound* is a sound synthesis language developed by Barry L. Vercoe at the Experimental Music Studio of the MIT Media Laboratory, and is freely available for educational and research purposes from `ftp://ftp.musique.umontreal.ca/pub/mirrors/dream`.

the obvious way. The segment generators indicated in the figure generate line-segment envelopes by linearly interpolating between the values of their respective first parameter streams with breakpoints at times specified by the second parameter stream; breakpoint times are the same for all four segment generators. The phase generator at the top left corner of the figure generates the step values for the table-lookup oscillator, as discussed in §1.1.3; all wave tables in the bank are scanned with the same phase values, since a single weighted average frequency differential is calculated for each breakpoint.

3.2 The Breakpoint Matching Algorithm

The breakpoint matching algorithm which was the primary focus of this research was designed to find a globally optimal set of weighted wavetable matches across all breakpoints, taking into account the requirement that wavetables must be faded in and out at the beginning and end of each span of use, respectively.

The result is a three-stage process, as illustrated in Figure 3.8. Each of these stages will be discussed in detail after an initial section which explains how the quality of a match is assessed and how the wavetable weightings are determined.

3.2.1 Match Evaluation

As Horner, Beauchamp, and Haken have shown in [44], the problem of determining the weightings (amplitude factors) of a set of basis spectra which provide the best match to a particular spectrum in a least-squares sense is a linear problem, and can be solved using matrix arithmetic. More particularly, it is an instance of the *general linear least squares problem* [74, §15.4], and can be solved by use of the *normal equations* [16, 74].

In the case of multiple wavetable matching, if it were possible to find a set of weightings (coefficients) of M basis spectra (wavetables) such that the weighted sum exactly matched the spectrum at a given breakpoint, this model could be represented as the matrix equation

$$\mathbf{A} \cdot \mathbf{c} = \mathbf{b} \tag{3.1}$$

where A is an $N \times M$ matrix, each column of which represents one of the M basis spectra as a column vector of its N harmonic amplitudes; c is a column vector of the M coefficients which will be used as weightings of the M wavetables; and b is a column vector consisting of the N harmonic amplitudes of the breakpoint spectrum to be matched. The equation above can thus be

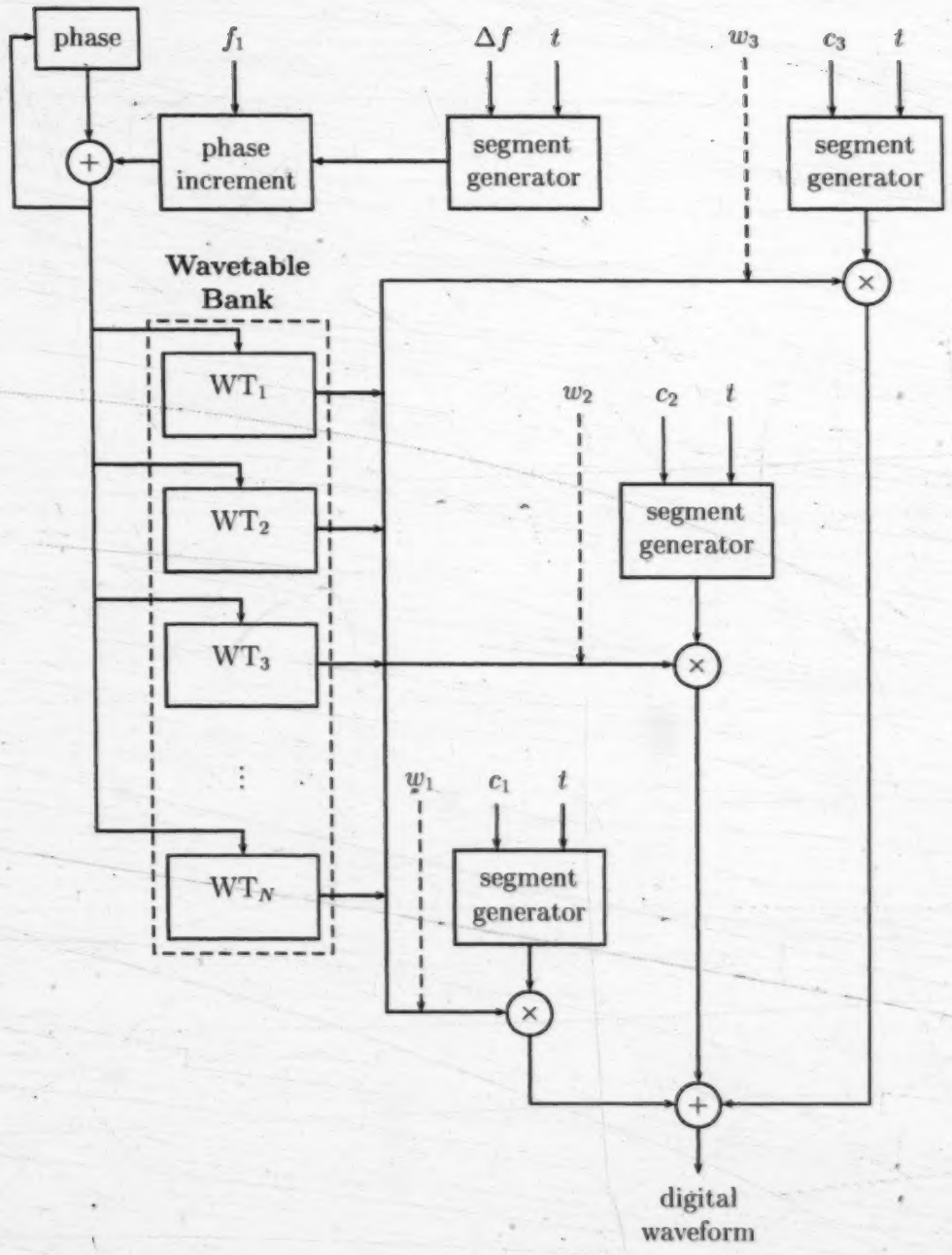


Figure 3.7: The optimized multiple wavetable interpolation synthesis model. A 3-oscillator synthesizer with an N-table wavetable bank is illustrated, where f_1 is the fundamental frequency, and t, Δf, w_i, and c_i are streams of breakpoint times, frequency differentials, wavetable indices, and wavetable coeffients (weightings), respectively. The dashed arrows indicate selection, by index, of a particular wavetable from the wavetable bank.

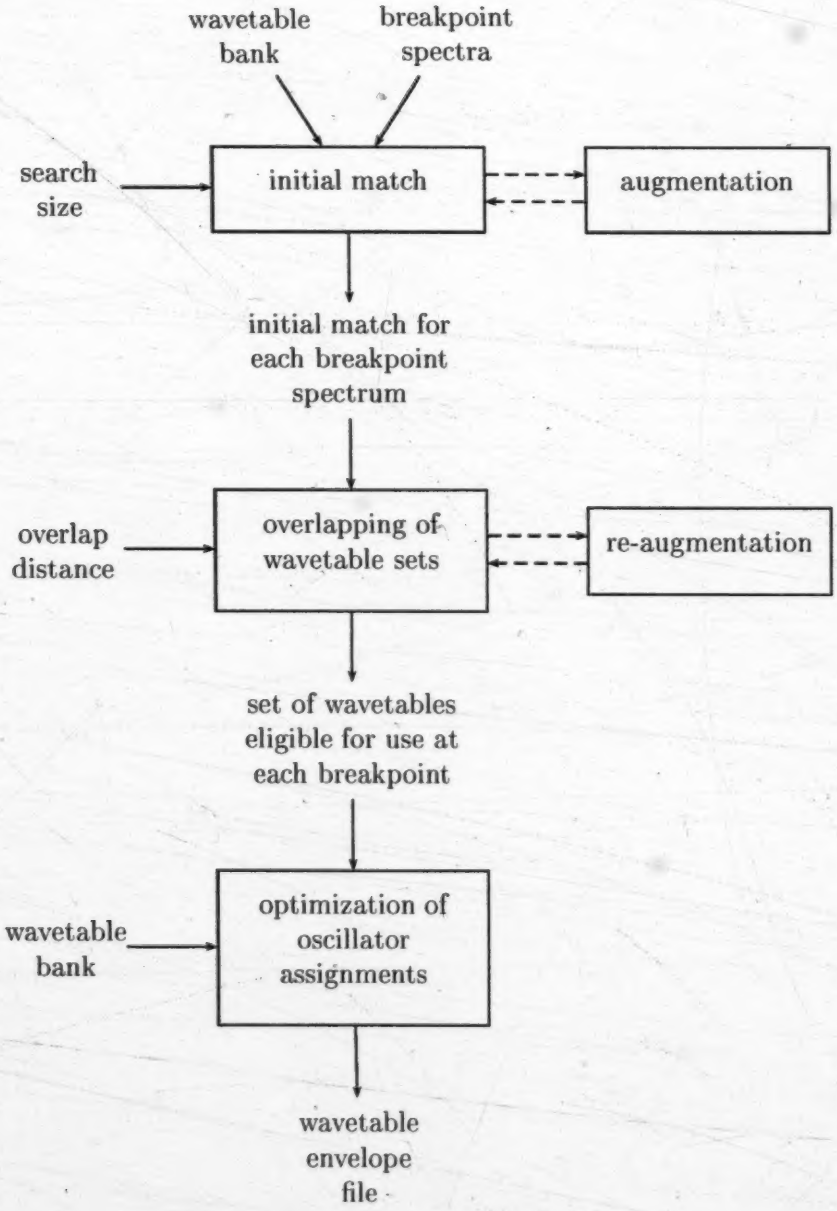


Figure 3.8: The stages of the globally optimal breakpoint matching algorithm.

restated as

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,M} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,M} \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \tag{3.2}$$

where $a_{i,j}$ is the amplitude of the ith harmonic of the jth wavetable (basis sprectrum), c_i is the coefficient (weight) to be applied to the ith wavetable, and b_i is the amplitude of the ith harmonic of the breakpoint spectrum to be matched. The problem can be stated equivalently as a linearly weighted sum:

$$\sum_{j=1}^{M} a_{i,j} c_j = b_i. \tag{3.3}$$

However, because it is desirable to use as few basis spectra as possible, $N > M$, so the system is overdetermined and generally does not have an exact solution. Instead, we seek to find the least squares solution of the system, that is, to minimize the squared error

$$\chi^2 = \sum_{i=1}^{N} \left(\sum_{j=1}^{M} a_{i,j} c_j - b_i \right)^2. \tag{3.4}$$

The minimum of (3.4) occurs where the derivative of χ^2 with respect to all M coefficients c_k is 0:

$$\sum_{i=1}^{N} 2 \left(\sum_{j=1}^{M} a_{i,j} c_j - b_i \right) a_{i,k} = 0. \tag{3.5}$$

The M equations (3.5) for $k = 1, 2, \ldots, M$ are called the *normal equations* of the least squares problem [74, p. 673]. They can be written in matrix form as

$$(\mathbf{A} \cdot \mathbf{c} - \mathbf{b})^T \cdot \mathbf{A} = 0 \tag{3.6}$$

or, equivalently [16, p. 770],

$$(\mathbf{A}^T \cdot \mathbf{A}) \cdot \mathbf{c} = \mathbf{A}^T \cdot \mathbf{b}. \tag{3.7}$$

This system can be solved by finding an LUP decomposition of $(\mathbf{A}^T \cdot \mathbf{A})$. Since

$$\mathbf{L} \cdot \mathbf{U} = \mathbf{P} \cdot \mathbf{A}^T \cdot \mathbf{A}, \tag{3.8}$$

this is equivalent to

$$\mathbf{L} \cdot \mathbf{U} \cdot \mathbf{c} = \mathbf{P} \cdot \mathbf{A}^T \cdot \mathbf{b} \tag{3.9}$$

which is easily solved by solving the two triangular linear systems

$$\mathbf{L} \cdot \mathbf{x} = \mathbf{P} \cdot \mathbf{A}^T \cdot \mathbf{b} \tag{3.10}$$

and

$$\mathbf{U} \cdot \mathbf{c} = \mathbf{x} \tag{3.11}$$

by forward and back substitution, respectively [16, pp. 751–753].

Once the LUP decomposition of $\mathbf{A}^T \cdot \mathbf{A}$ has been found, it can be used along with the permutation vector $\mathbf{P}$ and $\mathbf{A}^T$ with any number of different vectors $\mathbf{b}$ on the right-hand side of (3.9). As applied in this research, this means that the LUP decomposition of the transpose-product $\mathbf{A}^T \cdot \mathbf{A}$ where $\mathbf{A}$ is a particular set of wavetables (basis spectra) selected from the wavetable bank is calculated only once per selection and used multiple times to evaluate how well that set of wavetables can approximate the spectrum at each breakpoint or at some subset of the breakpoints.

3.2.2 The Initial Match

In the first stage of the breakpoint matching algorithm, an initial match of a user-specified size is found for each breakpoint. The size of the match (that is, the number of different wavetables used in the match) can maximally be the same as the number of oscillators to be used in the synthesis stage, but may also reasonably be less than the number of oscillators, since the set of wavetables to be considered for final use at a given breakpoint may be augmented with additional tables in later stages of the matching algorithm.

Two different methods were evaluated for use at this stage: a multi-level pruned search and a genetic algorithm.

Multi-level search

The best possible match at each breakpoint would be found by an exhaustive search of all $\binom{N_{wt}}{N_{osc}}$ combinations of wavetables selected N_{osc} at a time, where N_{osc} is the number of oscillators, from a wavetable bank of size N_{wt}. However, the cost of such a search becomes prohibitive for more than 3 or 4 oscillators (depending on the size of the wavetable bank). Furthermore, finding the best possible match of size N_{osc} for each breakpoint spectrum at this stage of the algorithm does not necessarily produce the best final result, as will be shown in Chapter 5, since, as discussed in §3.1 above, it is necessary to fade out a wavetable which ceases to be used from one breakpoint to the next or to fade in one which comes into use; as a result of this requirement, a set of matches to breakpoint spectra which has greater consistency—that is, which uses many of the same wavetables over a number of consecutive breakpoints—may lead to a better overall result than a set of matches with high specificity but greater variety of wavetable usage.

One way to reduce the cost of a search is to focus the search by pruning the search tree. This can be done in the present case by performing an exhaustive search for the best matches of some size less than N_{osc} and then extending the first-level search by a second level which seeks to augment only those sets of wavetables which provided a best match to at least one breakpoint spectrum in the first-level search. For example, if an eventual 4-oscillator match is desired, the search performed at this stage could search for the best 3-wavetable matches in the first level and augment those sets with a fourth wavetable in the second-level search (a "3+1" search). Alternatively, the first-level search could seek only 2-wavetable matches which would be augmented with two additional wavetables (a "2+2"search) or even a single additional wavetable (a "2+1" search) in the second-level search.

Figure 3.9 illustrates how a "2+1" search might prune a search tree. For simplicity, the tree diagram assumes that there are only six wavetables in the wavetable bank, and that the first-level, depth-2 search finds that only four different wavetable sets are used as best matches to the breakpoint spectra of the tone to be matched: {1, 3}, {2, 3}, {2, 6}, and {4, 5}. The second-level, depth-1 augmenting search explores the third level of the search tree only for these four subtrees.

The reference to a *multi-level* search (as opposed to *bi-level*) at this stage is based on the fact that, should the matches found at this stage be less than N_{osc} in size, even after the augmenting due to overlapping which will be discussed in reference to the next stage of the breakpoint matching algorithm, an additional level (or, in some cases, multiple levels) of search will be conducted to ensure that all breakpoint matches are of size N_{osc} prior to the final stage of the algorithm. It is for this reason that a "2+1" search might reasonably be specified at this stage even if a 4-oscillator match is ultimately desired.

As will be demonstrated in Chapter 5, a "3+1" search executes about an order of magnitude faster than a "4+0" search, yet yields about the same or better error rates, on average, after optimization. A "2+1" search is another order of magnitude faster than a "3+1" search, at the cost of an increased average error of about 50%; however, after optimization, the difference in average matching error is reduced to between 2% and 5%.

There are many examples to demonstrate that the set of wavetables selected by a first-level search of size n to match a particular breakpoint spectrum is often *not* a superset of the wavetables selected by a size $n-1$ search. For example, while adding one wavetable to a match of size 2 almost always improves the quality of the match (unless a perfect match has already been found, such as when the breakpoint spectrum being matched is one of the basis spectra in the wavetable bank), a significantly better match may be found by an exhaustive search of all matches of size 3, and the latter match may have few or no wavetables in common with the former matches.

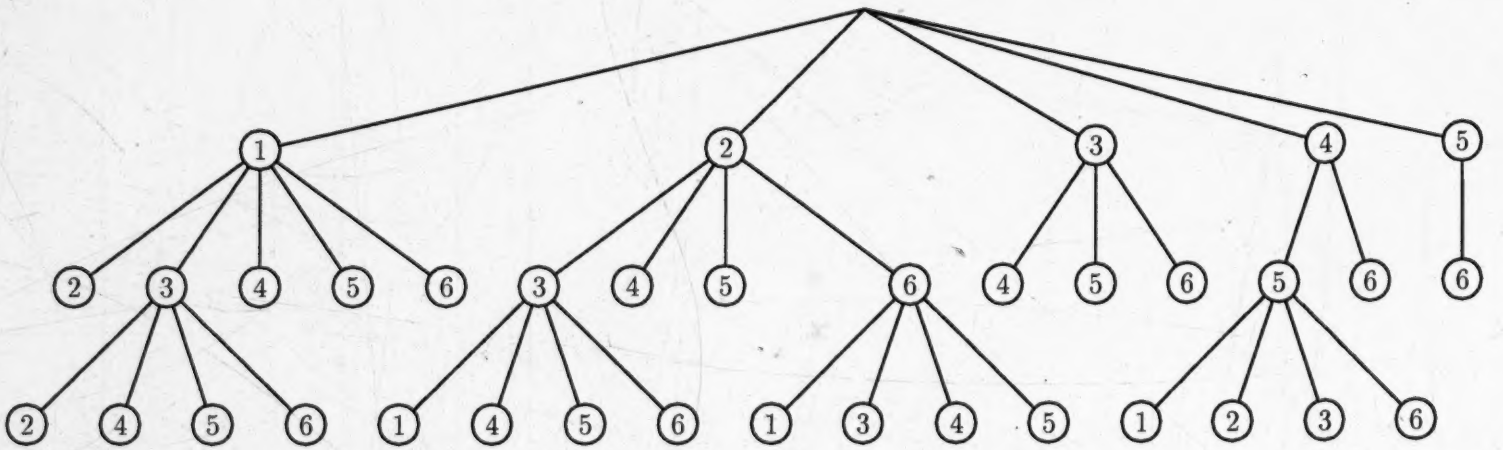


Figure 3.9: Search tree pruning by a "2+1" multi-level search, assuming that there are only six wavetables in the wavetable bank and that only the wavetable sets {1, 3}, {2, 3}, {2, 6}, and {4, 5} were found to be the best 2-table match to some breakpoint spectrum.

Genetic algorithm

The observations above regarding the unpredictable nature of various types of multi-level searches suggested that a genetic algorithm might perform well on this task, perhaps yielding matches about as good as a pruned search in less time.

A genetic algorithm (GA) is a form of probabilistic search which is guided by a strategy based on genetic inheritance. The implementor of the GA decides on a representation of a solution to the problem at hand (e.g., as a bit string or an array of floating point values) and provides an *objective function* which the GA can call to evaluate the *fitness* (quality) of a potential solution.[2] The GA begins by randomly generating and evaluating a number of potential solutions to the problem (each called an *individual* and collectively called a *population*). It then repeatedly generates a new *generation* of the population from the existing population by probabilistically "breeding" new individuals from pairs of existing individuals (the *parents* of the new *child*); individuals are typically selected from the population for reproduction in proportion to their fitness so that the quality of the population is likely to increase (*evolve*) over the generations. A new individual is creating by copying some portion of its *genome* (its digital representation) from one parent and the remainder from the other parent by a process called *crossover*; *offspring* are typically created in pairs and, in the simplest type of GA, replace their parents in the next generation. One or more of the best (most "fit") individuals from the current generation may be retained in the next generation by an option called *elitism*. Occasionally, an individual is changed slightly (e.g., by flipping one bit in its binary representation) by *mutation*. The genetic algorithm is typically terminated after a specified number of generations or when some level of convergence of the population or the fitness scores of the best individuals has been reached.

A genetic algorithm was implemented as an alternative to the first level of the multi-level search discussed above; that is, if an initial search of size N_{osc} was specified, the genetic algorithm was used instead of exhaustive search, but if an initial search of size less than N_{osc} was specified, a second-level pruned search could be used to augment the matches found by the genetic algorithm.

Testing of this approach confirmed that, when used as a first-level search in combination with an exhaustive second-level search, the genetic algorithm could find matches about as good as those found by exhaustive or pruned search, but savings in time were only realized relative to the larger exhaustive searches, and then only if the objective function cached the results of calls to the LUP decomposition and least squares evaluation functions (as discussed in §4.4.2, page 79).

[2]The GA may not use the value returned by the objective function directly: fitness scores are typically derived by some form of scaling of objective scores.

u:	5	8	13
v:	5	8	13
w:	5	8	21
x:	5	8	13
y:	5	8	13

(a) The initial match.

u:	5	8	13
v:	5	8	—
w:	5	8	21
x:	5	8	—
y:	5	8	13

(b) One possible oscillator assignment.

u:	5	8	13
v:	5	8	13
w:	5	8	—
x:	5	8	13
y:	5	8	13

(c) Another possible oscillator assignment.

Figure 3.10: A possible sequence of wavetable matches and two possible optimized oscillator assignments. The letters $u \ldots y$ are used as breakpoint identifiers instead of actual breakpoint numbers because this is a hypothetical example. The dashes indicate points at which wavetables are faded out and in.

3.2.3 Overlapping of Wavetable Sets

The second stage of the breakpoint matching algorithm is intended to provide more flexibility in the subsequent optimization stage which, as part of its task of assigning wavetables to oscillators, must decide when to fade a wavetable in or out of use. For example, the initial matches for some consecutive set of breakpoint spectra might include a wavetable which passes out of use and then back into use, as illustrated in Figure 3.10.

In this case, the optimizer is likely to pick one of the two assignments of wavetables to oscillators shown in Figure 3.10, parts (b) and (c), where the dash indicates the point at which the wavetable assigned to the relevant oscillator at the previous breakpoint has been faded out to zero amplitude and a new wavetable—the one assigned to this oscillator at the next breakpoint—begins a fade-in from zero amplitude. If the use of wavetable 21 at breakpoint w results in a lower overall error level than the use of wavetable 13 at both breakpoints v and x, then the first option will be used; otherwise, the latter option will be preferred. (It is assumed here that wavetables 5 and 8 are very important in achieving a good match at all five breakpoints, since they appear in all five matches.)

However, it is possible—indeed, highly likely—that an even lower overall error level would be achieved by allowing the optimizer to use wavetable 13 instead of 21 at breakpoint w: the higher error resulting from substituting wavetable 13 for 21 at breakpoint w in Figure 3.10(b) will likely be offset by the lower error levels achieved by using three-wavetable matches instead of two-wavetable matches at both breakpoints v and x, and a three-wavetable match will surely be better than a two-wavetable match at breakpoint w in option (c).

The optimizer could be tuned to look for special cases like this through a look-ahead or look-behind routine, but a more general solution to the problem

u:	5	8	13		u:	5	8	—		u:	5	8	—
v:	5	8	—		v:	5	8	21		v:	5	8	21
w:	5	8	21		w:	5	8	21		w:	5	8	21
x:	5	8	21		x:	5	8	—		x:	5	8	21
y:	5	8	—		y:	5	8	13		y:	5	8	—
	(a)					(b)					(c)		

Figure 3.11: Other possible optimized oscillator assignments with one-breakpoint overlapping.

of selecting the best points for fade-in and fade-out of wavetables can be expected to yield even better results. For example, it could be the case that the use of wavetable 21 at breakpoint w is crucial to achieving a low error measure at that breakpoint but that a two-wavetable match would be more tolerable at breakpoint y than at breakpoint x. If the optimizer were allowed to use *any* wavetable one or more breakpoints earlier or later than suggested by the initial sequence of matches, then any of the scenarios depicted in Figure 3.11 would also be possible and may provide a lower overall error than either of the options in Figure 3.10.

The general solution adopted for this breakpoint-matching algorithm is to include a stage in which wavetable sets are overlapped with the wavetable sets at preceding and following breakpoints before the optimizer makes oscillator-assignment decisions in the following stage. A minimum overlap of one breakpoint is mandated, but overlapping at any distance may be specified; the default overlap distance is two breakpoints on either side of the breakpoint which is the focus of the overlapping algorithm. The greater the overlap distance, the more possibilities which must be evaluated by the optimizer, so it is best to limit the amount of overlap to distances from one to three.

A specific example will illustrate both the general idea of overlapping and some of its implications. Figure 3.12(a) shows the first 30 harmonics of the phase vocoder analysis of a French horn tone at pitch E2, and part (b) shows the corresponding 24-breakpoint piecewise linear approximation found by the breakpoint selection algorithm. Figure 3.13(a) shows the best three-wavetable matches to the breakpoint spectra as found by a first-level search of depth two augmented by an second-level search of depth one (a "2+1" search). Note that breakpoints 7 to 14 were best matched by the same set of wavetables. As a result, as shown in Figure 3.13(b), the wavetable sets at breakpoints 9 to 12 which would have been passed to the optimizer after overlapping at distance two were of size three, one less than the number of oscillators to be used for resynthesis. A better result would undoubtedly be achieved by re-augmenting the wavetable sets at breakpoints 9 to 12 to size four, since even if one oscillator is used between breakpoints 8 and 9 to fade out wavetable 6 and again between

breakpoints 12 and 13 to fade in either wavetable 4 or 23, the oscillator would be available to use a fourth weighted wavetable at breakpoints 10 and 11.

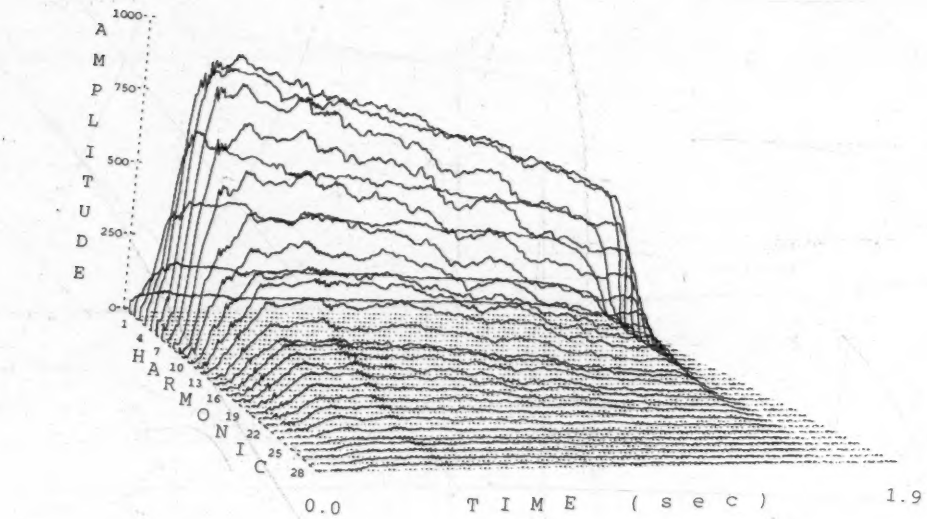
Figure 3.13(c) shows the best matches that were found by re-augmentation of the wavetable set {10, 15, 37}. Although the re-augmentation phase—a third-level exhaustive search—was triggered by the existence at breakpoints 9 to 12 of a wavetable set smaller in size than the number of oscillators to be used, the search tries matching every possible combination of wavetables 10, 15, 37, and one other wavetable at every breakpoint. If the new combination is a better match at any breakpoint, overlapping is repeated for that breakpoint using the new set of wavetables. In the example shown, it is to be expected that a four-wavetable match including wavetables 10, 15, and 37 will improve the matches at breakpoints 7 to 14 and 16, since they already have the set {10, 15, 37} as the best three-table match; note that at some of the breakpoints, the match was improved by the addition of wavetable 6, at others, with 27, and at breakpoint 16, with 41. In addition, improved matches were found for breakpoints 5, 6, 15, 17–19, 21 and 22, and overlapping was performed with the new four-wavetable sets for these breakpoints as well, resulting in the final sets of potentially usable wavetables shown in Figure 3.13(d). Note that the overlapping which was performed during the first pass centered on the latter breakpoints is not undone during the second pass; this leaves the optimizer with more options for accommodating fade-in and fade-out, since it may be, for example, that using the three-table match {6, 9, 15} at breakpoint 3 will give a better overall result than using the new four-table match.

3.2.4 Optimization of Oscillator Assignments

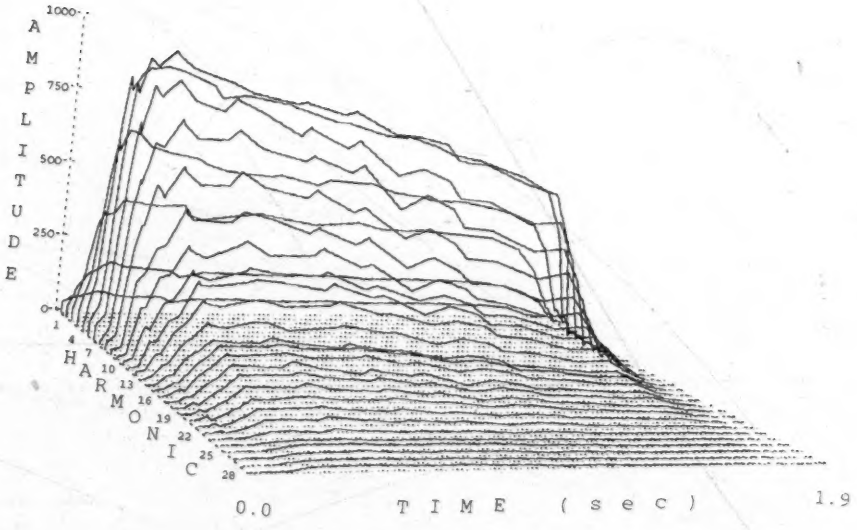
The task of the final phase of the breakpoint-matching algorithm is to assign a weighted wavetable to each available oscillator at each breakpoint such that the overall error is minimized, taking into account the need to fade a wavetable in or out when it begins or ceases to be used. A synthesizer that subsequently uses this oscillator-assignment data to synthesize a tone, any processing algorithms that convert this data to another form for synthesis (such as to a *Csound* file), or a composer who wishes to modify the control data to create a modified tone must recognize that a wavetable with a weight of zero represents not only the start of the fade-in of a new wavetable, but that the previous wavetable assigned to the same oscillator at the previous breakpoint should be faded out between the previous breakpoint and the current one. If wavetable zero is specified (i.e., zero appears in place of a valid wavetable index) for an oscillator, then that oscillator is unassigned (unused) between the current breakpoint and the next. (Due to optimization, this occurs almost exclusively at the final, external breakpoint only.)

The optimization of oscillator assignments is achieved by modeling the problem as a vertex-weighted directed acyclic graph (DAG) and using the single-source acyclic weighted shortest path algorithm [16, §25.4] [93, §14.5].

A DAG is constructed such that each vertex represents a particular wave-



(a) Phase vocoder analysis (first 30 harmonics).



(b) A 24-breakpoint piecewise linear approximation.

Figure 3.12: Analysis of a French horn tone at pitch E2.

1:	10	26	29
2:	6	9	15
3:	6	9	15
4:	4	15	21
5:	4	15	47
6:	6	15	37
7:	10	15	37
8:	10	15	37
9:	10	15	37
10:	10	15	37
11:	10	15	37
12:	10	15	37
13:	10	15	37
14:	10	15	37
15:	4	15	23
16:	10	15	37
17:	6	15	37
18:	4	15	23
19:	4	15	23
20:	4	15	27
21:	4	10	15
22:	4	15	37
23:	9	10	30
24:	16	45	46

(a) Initial "2+1" matches.

1:	{ 6 9 10 15 26 29 }
2:	{ 4 6 9 10 15 21 26 29 }
3:	{ 4 6 9 10 15 21 26 29 47 }
4:	{ 4 6 9 15 21 37 47 }
5:	{ 4 6 9 10 15 21 37 47 }
6:	{ 4 6 10 15 21 37 47 }
7:	{ 4 6 10 15 37 47 }
8:	{ 6 10 15 37 }
9:	{ 10 15 37 }
10:	{ 10 15 37 }
11:	{ 10 15 37 }
12:	{ 10 15 37 }
13:	{ 4 10 15 23 37 }
14:	{ 4 10 15 23 37 }
15:	{ 4 6 10 15 23 37 }
16:	{ 4 6 10 15 23 37 }
17:	{ 4 6 10 15 23 37 }
18:	{ 4 6 10 15 23 27 37 }
19:	{ 4 6 10 15 23 27 37 }
20:	{ 4 10 15 23 27 37 }
21:	{ 4 9 10 15 23 27 30 37 }
22:	{ 4 9 10 15 16 27 30 37 45 46 }
23:	{ 4 9 10 15 16 30 37 45 46 }
24:	{ 4 9 10 15 16 30 37 45 46 }

(b) Wavetable sets with overlap of 2.

3:	4	10	15	37
5:	4	10	15	37
6:	6	10	15	37
7:	6	10	15	37
8:	10	15	27	37
9:	6	10	15	37
10:	6	10	15	37
11:	10	15	27	37
12:	10	15	27	37
13:	10	15	37	41
14:	6	10	15	37
15:	10	15	37	41
16:	10	15	37	41
17:	10	15	37	41
18:	10	15	37	41
19:	10	15	37	41
21:	4	10	15	37
22:	4	10	15	37

(c) Re-augmented matches.

1:	{ **4** 6 9 10 15 26 29 **37** }
2:	{ 4 6 9 10 15 21 26 29 **37** }
3:	{ 4 6 9 10 15 21 26 29 **37** 47 }
4:	{ 4 6 9 **10** 15 21 37 47 }
5:	{ 4 6 9 10 15 21 37 47 }
6:	{ 4 6 10 15 21 **27** 37 47 }
7:	{ 4 6 10 15 **27** 37 47 }
8:	{ 6 10 15 **27** 37 }
9–10:	{ **6** 10 15 **27** 37 }
11–12:	{ **6** 10 15 **27** 37 **41** }
13–14:	{ 4 **6** 10 15 23 **27** 37 **41** }
15–17:	{ 4 6 10 15 23 37 **41** }
18–19:	{ 4 6 10 15 23 27 37 **41** }
20:	{ 4 10 15 23 27 37 **41** }
21:	{ 4 9 10 15 23 27 30 37 **41** }
22:	{ 4 9 10 15 16 27 30 37 45 46 }
23–24:	{ 4 9 10 15 16 30 37 45 46 }

(d) Wavetable sets after overlap of re-augmented matches (with changes highlighted).

Figure 3.13: Two-breakpoint overlapping with re-augmentation for a four-oscillator match of a French horn playing pitch E2.

table set at a particular breakpoint and each edge represents a legal transition to some wavetable set at the next breakpoint, taking into consideration the need to fade out a wavetable which ceases to be used between the current breakpoint and the next and to fade in a wavetable which starts to be used at the next breakpoint. The weight (cost) of each vertex is the least-squares error of the fit of the weighted wavetables to the breakpoint spectrum. The shortest path algorithm is invoked with the representation of the external breakpoint at the start of the tone as the source vertex. The path to the vertex representing the external breakpoint at the end of the tone is then traced; the sequence of vertices on the shortest path from the start vertex to the end vertex represents the globally optimal sequence of sets of wavetables, one set per internal breakpoint.

The wavetables of each set are then assigned to oscillators so as to ensure continuity of wavetable assignments from one breakpoint to the next. Figure 3.14 shows the result of match optimization and wavetable-to-oscillator assignment for the same French horn tone used as an example in Figures 3.12 and 3.13.

The running time of the single-source acyclic weighted shortest path algorithm is $\Theta(V + E)$ if the graph is implemented with an adjacency list representation. However, the size of the graph is not simply determined by the size of the initial best match to each breakpoint spectrum nor by the number of oscillators to be used in synthesis. Because the number of vertices is determined by the number of combinations of wavetables drawn from the set of eligible wavetables at each breakpoint, the size of the graph is related to the complexity of the tone being analyzed. As can be seen in Figures 3.12 and 3.13, the wavetable sets from which optimal matches will be selected are larger in the attack and release phases of the tone and smaller in the sustain portion; the more consistent the spectral envelope is over a sequence of breakpoints, the smaller the wavetable sets for those breakpoints will be. The graph used to optimize the breakpoint matches for the horn E2 tone which was the subject of those figures had 3,590 vertices and 101,366 edges, or 4,373.2 vertices and edges per breakpoint. This is about half the average size of graphs used in finding four-oscillator matches with an initial "2+1" search and less than one-fifth the size of the largest graphs resulting from the same type of search with the same wavetable bank; on a per-breakpoint basis, this graph was about two-thirds the size of the average graph size and one-fifth the size of the largest graphs.

There is a general relationship, however, between the average number of vertices and edges per breakpoint and the specificity of the initial search for best matches. The graph resulting from an initial "3+1" search is three to four times as large, per breakpoint, as the graph constructed from the results of an initial "2+1" search and any re-augmentation which may have been triggered in generating four-oscillator matches; a graph derived from an initial "4+0" search is, on average, about one-quarter larger per breakpoint than one resulting from a "3+1" search.

Break-point	Oscillator 1	2	3	4
1:	4	10	15	37
2:	4	10	15	37
3:	4	10	15	37
4:	4	10	15	37
5:	4	10	15	37
6:	4	10	15	37
7:	4	10	15	37
8:	—	10	15	37
9:	6	10	15	37
10:	6	10	15	37
11:	—	10	15	37
12:	41	10	15	37
13:	41	10	15	37
14:	41	10	15	37
15:	41	10	15	37
16:	41	10	15	37
17:	41	10	15	37
18:	41	10	15	37
19:	41	10	15	37
20:	41	10	15	—
21:	—	10	15	4
22:	37	—	15	4
23:	37	16	15	4
24:	37	16	15	4

Figure 3.14: Oscillator assignment for a four-oscillator match of a French horn playing pitch E2.

The relationship between specificity and graph size may be understood by considering the *locality* of spectra in musical tones: a spectrum at a particular point in time of the tone's duration is most likely to be similar to neighboring spectra.[3] In the case of a tone played with vibrato, the tone is likely to alternate between two spectral localities; a spectrum near the peak of a vibrato is not only similar to the spectra near it, but is also likely to be similar to spectra near the preceding and following vibrato peaks. An initial exhaustive search for the best two-wavetable matches to breakpoint spectra is likely to find the same matches being used at adjacent or nearby breakpoints; however, the best four-wavetable matches might use a wide variety of wavetables because, as the matches to each spectrum get more exact, the particularities of the spectra become more significant than their similarities.

The use of the single-source acyclic weighted shortest path algorithm to find a globally optimal set of wavetable matches to breakpoint spectra is predicated on the use of some kind of initial search to find a good match for each breakpoint spectrum—for identifying localities—and on the overlapping of these matches to form sets of wavetables eligible for use at each breakpoint to allow smooth transitions from locality to locality.

The sequence of matches found by the shortest path algorithm is globally optimal in a mathematical sense, but may not provide the best sequence of matches from a perceptual point of view. For example, several studies [8, 38, 40, 82] have found that the attack portion of a tone has high perceptual relevance, so smaller errors in matching spectra in the attack portion may be more perceptually significant than larger errors in the sustain or release portions. However, as previously stipulated, the algorithm is globally optimal with respect to a given error measure; an error measure could include a higher weighting for errors in the attack portion of a tone.

The construction of the DAG through which the shortest path is to be found is based on an algorithm which adds an edge to the graph representing each possible transition from one possible match at one breakpoint to another possible match at the next breakpoint, ensuring that the requirements for fade-in and fade-out of wavetables is taken into account. This algorithm is discussed in §4.4.4.

As is the case with almost any algorithm, there are trade-offs to be made between the time expended in seeking a result and the quality of the result found, and the breakpoint-matching algorithm presented here is no different. Chapter 5 will present the results of applying this algorithm with various options and will make recommendations concerning which trade-offs might be appropriate in different situations.

[3] Sandell and Martens [83] found high correlation between neighboring spectra and that the correlation between spectra was greater than the correlation between the temporal envelopes of harmonics.

Chapter 4

Implementation Issues

4.1 Waveform Analysis

Instrumental tones were analyzed using the **pvan** (phase vocoder analysis) utility of the *SNDAN* sound analysis suite by James Beauchamp and his students at the University of Illinois at Urbana-Champaign [5]. Details of the operation of **pvan** are given elsewhere [5, 59], but, in summary, **pvan** is a period-synchronous phase vocoder which, given an estimated or calculated (average) frequency of analysis, uses band-limited interpolation of the input signal to produce a power-of-two number of samples per period, a Hamming window with a length equal to two periods, overlapping of analysis frames at a distance of one-half a period, and STFT analysis to produce an analysis file containing amplitude and frequency-deviation data for each harmonic, where the number of harmonics is limited by the Nyquist frequency.

While extensions of the McAulay-Quatieri (MQ) analysis technique [60] have been preferred by several researchers [28, 44, 57] for its ability to track large frequency deviations, the MQ method proved to be less useful than the phase vocoder in this research, just as it has in some other contexts [6]. First, it is less robust than the phase vocoder, in the sense that it requires the adjustment of several parameters to avoid analysis artefacts; in some cases using the **mqan** utility also available in the *SNDAN* suite, analysis artefacts resulted no matter which parameters were used, especially in the lower-amplitude attack and decay sections. Second, multiple wavetable synthesis requires that all partials be harmonic partials, since a single period of a waveform is stored in each wavetable, so the multiple tracks identified by MQ analysis must be reduced to harmonics (for which the *SNDAN* utility **harmformat** is provided); the result is cleaner than that of **pvan**, since small amplitude fluctuations (shimmer) are suppressed, but inharmonic partials cause crosstalk between harmonics. The problems of crosstalk due to inharmonic partials and decay artefacts and the fact that the vibrato of all the tones being used in testing was within the $\pm 2\%$ range over which the phase vocoder performs well made phase-vocoder analysis the preferred option.

4.2 Breakpoint Selection

Many piecewise-linear-approximation (PLA) algorithms have been proposed and reviewed for use in selecting breakpoints in musical tones, from the hand-selected line-segment approximations of Risset and Mathews [77] and Grey [38], and Beauchamp's *LINSEG* [2], through Strawn's testing [90] of Pavlidis's split-and-merge [68] and *ADJUST* [67] algorithms, to the genetic algorithm used by Horner and Beauchamp [43].

Breakpoint selection algorithms can in general be categorized according to whether they seek a PLA which conforms to specified error bounds or one which finds a specified number of breakpoints. Serra, Rubine and Dannenberg's *spectral-ramp interpolation* [86] is an example of the first type of algorithm, and operates in a similar fashion to *LINSEG*: starting at the beginning of the tone, they extend a spectral ramp to subsequent periods until the error exceeds a given threshold, at which point the previous period is stored as a breakpoint and the process is repeated to find the next ramp. Horner and Beauchamp's genetic algorithm for PLA [43] is an example of the second type: the genetic algorithm uses a bitstring or array of integers of a fixed size as its genome. Jensen's split-point time estimation method [50] is a recent and extreme example: it seeks only four breakpoints ("split-points"), assuming that the envelope of each partial of every tone corresponds to a variation of a prototypical attack-sustain-release (ASR) envelope.

Breakpoint-selection algorithms can also be classified according to whether they find a PLA for each partial or harmonic independently or whether they find shared (simultaneous, common) breakpoints. Both Serra's and Horner's algorithms seek shared breakpoints, such as are required for methods based on interpolation.

4.2.1 A Segment-Merging Algorithm

For this research, a new breakpoint selection algorithm was developed and used. The *segment-merging* algorithm can be used either to construct a PLA within a specified global error bound or with a specified number of breakpoints (or to halt on either condition). It is used here to find shared breakpoints, but could be modified for use on a single harmonic at a time. The algorithm proceeds in the opposite direction of Pavlidis's *SPLIT* algorithm: it begins with every analysis frame as a breakpoint (so the overall approximation error is zero) and, on every iteration, merges the two adjacent segments, the merging of which will increase the overall error the least. The algorithm halts when the number of breakpoints has been reduced to the user-specified target or when the error has increased beyond the user-specified threshold.

The segment-merging algorithm offers several advantages over the genetic algorithm which was found to be best in Horner and Beauchamp's comparison of PLA methods [43] and which was also tested:

- The segment-merging algorithm retains the extremes—the highest peaks

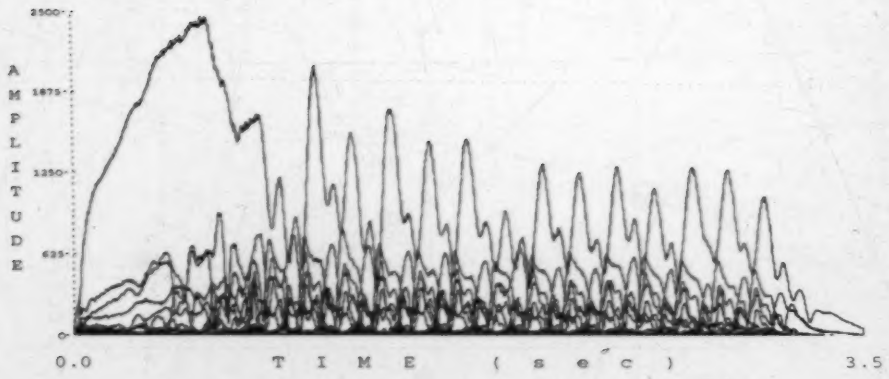
and the lowest valleys—as breakpoints. While this might result in higher overall error levels than methods which seek global optima rather than local ones, the more angular PLA's which result from segment merging are well suited to capturing both the sudden peaks of a strong attack and the regular oscillations of vibrato, and the tendency of the algorithm to exaggerate the profile of a tone with sudden changes in amplitude is compensated for by using more than the minimum number of breakpoints which have been found to be necessary through listening tests [43]. Figure 4.1 illustrates the advantages of segment-merging for approximating the envelope of a tone with vibrato; note that the PLA found by the GA (part b) shows the sixth major peak from the end of the original tone (part a) as lower in amplitude than its neighbors and misses the final peak entirely, while the segment-merging algorithm (part c) captures almost all the jaggedness of the original violin A4 tone.

- The segment-merging algorithm is predictable: the set of breakpoints found by a search for n breakpoints is a subset of those found by a search for m breakpoints, $m > n$. If visual inspection shows that an n-breakpoint PLA has missed one of the peaks of a vibrato, for example, one can try $n + 1$, $n + 2$, and so on until the peak is suitably modeled. Figure 4.2 shows that the breakpoints of the 55-breakpoint PLA found by segment merging (part c) are a subset of those of the 68-breakpoint PLA (part b) of the violin A4 tone (part a). It requires 55 breakpoints to capture the lower-amplitude vibrato peak near the middle of the tone and 68 breakpoints to capture the valley immediately after it.[1]

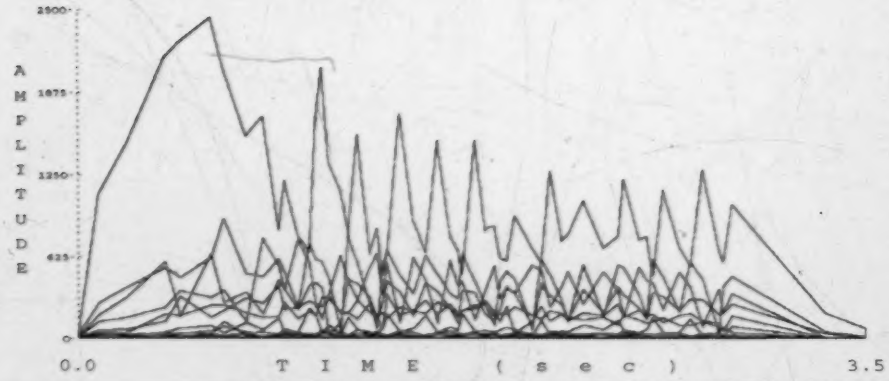
- The segment-merging algorithm is significantly faster than the genetic algorithm on this task, and finds better (i.e., lower-error) solutions. For example, on the violin A4 test case illustrated in Figures 4.1 and 4.2, the segment-merging algorithm took about 1 second for each of the 55- and 68-breakpoint PLA's, achieving mean error rates (as explained below) of about 150 and 125, respectively, while the GA took 691 seconds and 970 seconds, respectively, for the same task, achieving mean error levels of about 161 and 140.

As implemented (in C++) for this research, the segment-merging algorithm uses a locator-based priority queue to keep track of the segments of the current state of the PLA and, when the minimum element in the priority queue is retrieved and dequeued, to return the node corresponding to the segment which, when merged with the following segment, will increase the overall error the least. A *locator* is a design pattern introduced by Goodrich and Tamassia [36, §6.4] which keeps track of the current position of an element in a positional container. In a locator-based priority queue, the priority of an element may be

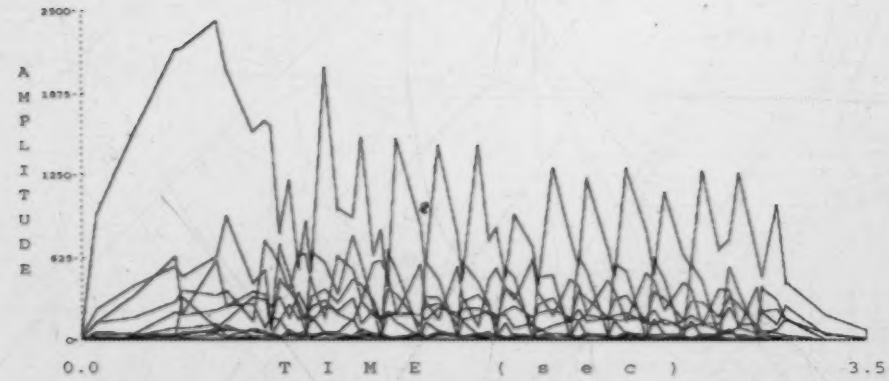
[1] This may suggest that the importance of modeling these features is more important to the human visual system than it might be to the human aural system; formal listening tests might be able to determine the salience of various details of harmonic amplitude envelopes.



(a) Phase vocoder analysis of the original tone.

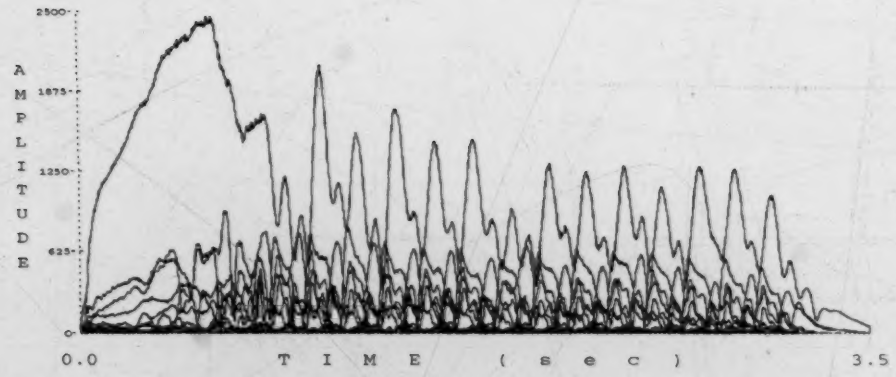


(b) PLA found by a genetic algorithm.

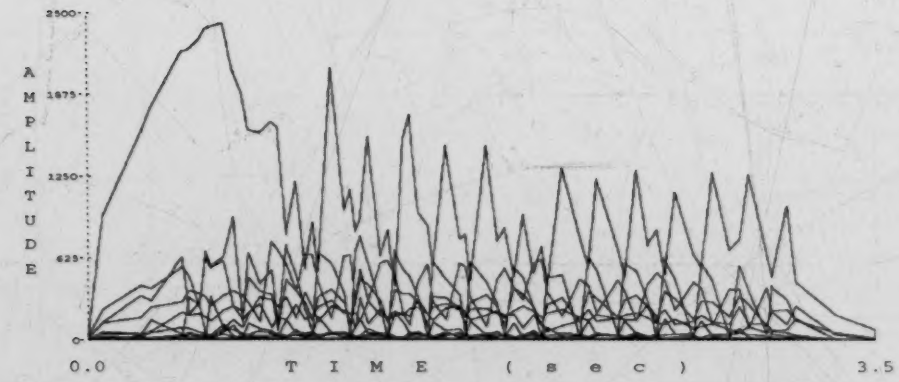


(c) PLA found by segment merging.

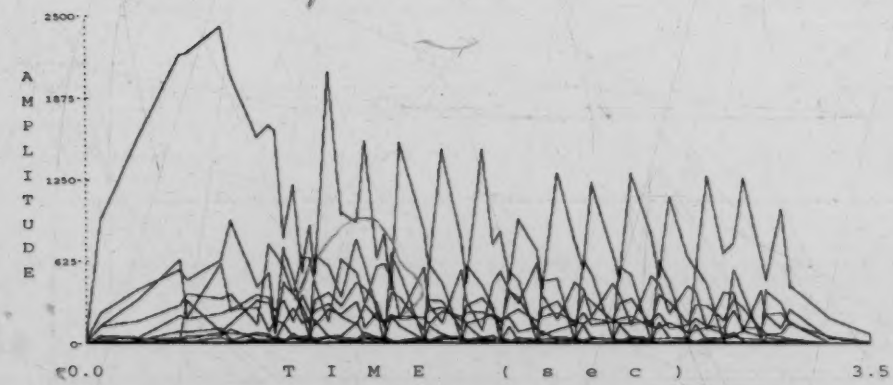
Figure 4.1: Fifty-five breakpoint PLA's of a violin tone at pitch A4 found by a genetic algorithm and by segment merging.



(a) Phase vocoder analysis of the original tone.



(b) Sixty-eight-breakpoint PLA.



(c) Fifty-five-breakpoint PLA.

Figure 4.2: Comparison of PLA's of different resolution of a violin tone at pitch A4 found by segment merging.

changed by accessing it through its locator; the locator then triggers an update of the priority queue, which not only changes the position of the element in the queue but updates the locator with the new position so that the association between the element and its position in the queue is maintained. A locator is needed in the implementation of the segment-merging algorithm because when a segment is dequeued from the priority queue and merged with the subsequent segment, the node in the queue representing the subsequent segment must be updated to represent the new merged segment; this will also cause the node to be moved to a new position in the priority queue because its priority—the amount by which merging this segment with the following segment will increase the overall error—will also have changed. Similarly, the segment preceding the dequeued segment will have to be updated, since its priority will also be changed by the merge.

4.2.2 Error Measures

The implementation uses an external error-measure object to evaluate the current state of the PLA. Two types of error measure were implemented and tested, with two variations of each type being available.

First, the user may select between evaluating the error as the Euclidean distance (root sum of squares error or L_2 norm) or the Manhattan (city-block) distance (sum of absolute magnitudes error or L_1 norm) between each actual spectrum and the corresponding spectrum of the linear approximation. These error measures are defined as

$$\text{Mean Euclidean error} = \frac{1}{N_{\text{frames}}} \sum_{n=1}^{N_{\text{frames}}} \sqrt{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2} \tag{4.1}$$

$$\text{Mean Manhattan error} = \frac{1}{N_{\text{frames}}} \sum_{n=1}^{N_{\text{frames}}} \sqrt{\sum_{k=1}^{N_{\text{har}}} |a_k(n) - a'_k(n)|} \tag{4.2}$$

where N_{frames} is the number of analysis frames, N_{har} is the number of harmonics to be used by the error measure, $a_k(n)$ is the amplitude of the kth harmonic at the nth frame, and $a'_k(n)$ is the corresponding amplitude from the linear approximation.

The implementation also allows the user to specify how many of the analysis harmonics are to be used to calculate the error. The error figures reported in Chapter 5 represent mean Euclidean error, calculated over all analysis harmonics.

Second, the user may specify the use of a relative error measure, a biased (or weighted) error measure, or both in combination. A relative error measure in effect divides the error of each frame by the RMS amplitude of the frame:

$$\text{Mean relative error} = \frac{1}{N_{\text{frames}}} \sum_{n=1}^{N_{\text{frames}}} \sqrt{\frac{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{\text{har}}} a_k(n)^2}} \tag{4.3}$$

This is the error measure preferred by Horner [41, 42, 44], presumably on the assumption that small amplitude differences will be more perceptible in low-amplitude portions of a tone than in high-amplitude portions, that is, that human hearing is relative rather than absolute.[2]

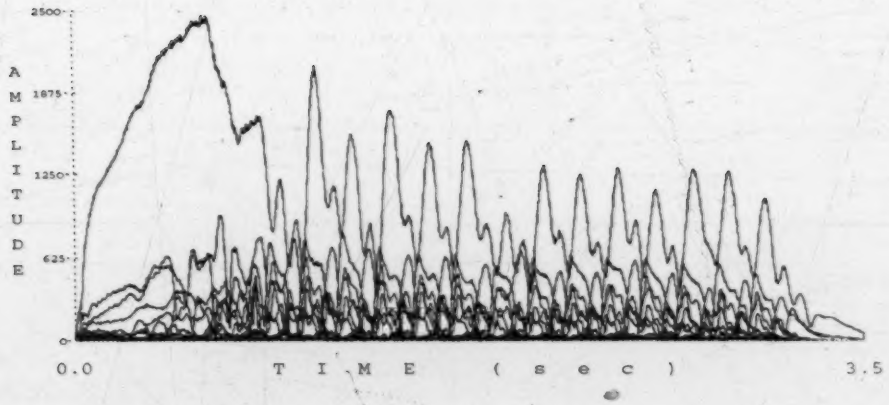
Relative error measures were tested but not used in the research reported here because it was found that they caused breakpoints to be selected which captured all the valleys in a tone with vibrato, but which missed some of the peaks. For example, Figure 4.3 shows a two-dimensional plot of the first 20 harmonics of a violin tone at pitch A4 and two 42-breakpoint PLA's, both found by the new segment-merging algorithm, the first using absolute Euclidean error, the second, relative Euclidean error. Note that the relative error measure misses a prominent peak about one-third of the way through the tone, but captures the small valley near the end of the tone; both the error measures miss a minor peak near the middle of the tone.

Another example of the problems with using a relative error measure is seen in Figure 4.4, which shows that the PLA constructed using a relative error measure devotes 17 of its 24 breakpoints to the tail of the decay portion of the tone (which may be room echo and is not audible in the original tone).

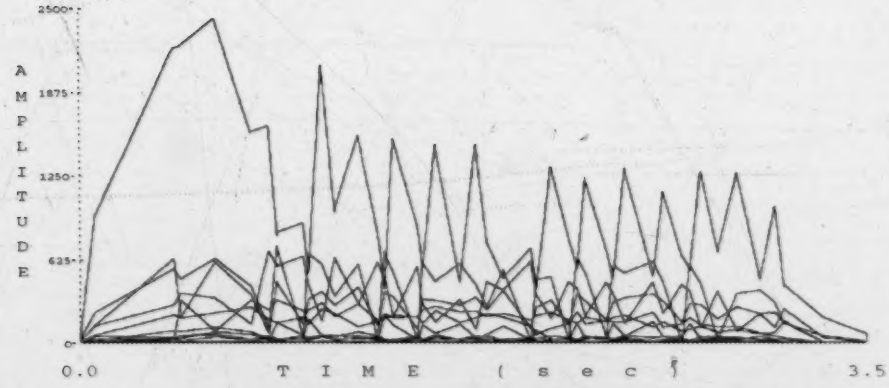
It is important that all major peaks and valleys of a tone with vibrato be captured in the PLA since the peaks and valleys of the frequency envelope correspond to the peaks and valleys of the amplitude envelope—in fact, the amplitude modulation is vibrato-induced—and the ear is very sensitive to the pitch fluctuations of vibrato [57]. Frequency modulation could be explicitly modeled by the breakpoint selection algorithm—the error measure could be modified to include the weighted average frequency differential or the frequency envelope could be modeled by a separate PLA—but the assumption that both amplitude and frequency envelopes will be well approximated by the same set of common breakpoints was found to be true in general.

A biased or weighted error measure gives more weight to errors in the perceptually significant attack segment of a tone. There are two ways in which the bias could be specified and applied, which might be called *segment-weighted* and *frame-weighted*. In the former option, a weight w_1 might be

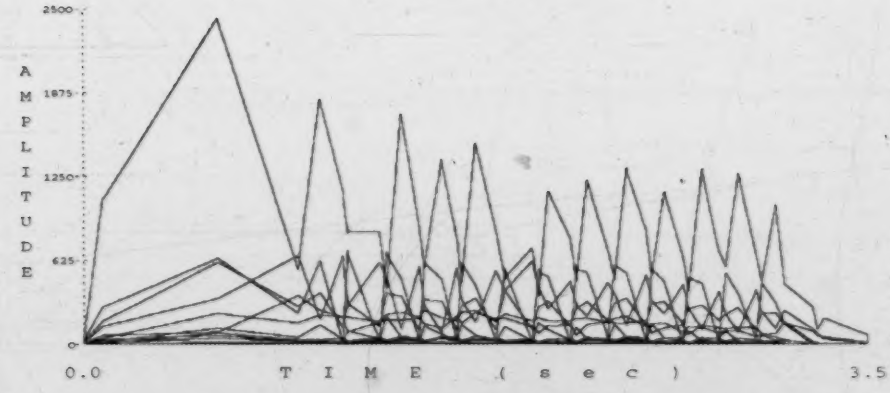
[2] Horner et al. explain [44, p. 340], "we would expect that the lower the value of this error measure, the better the perceptual match. Our experience so far is that this is generally but not always true. However, lacking a formula which is a good predictor of subjective preference, this is what we are using for the time being."



(a) Phase vocoder analysis of the original tone.

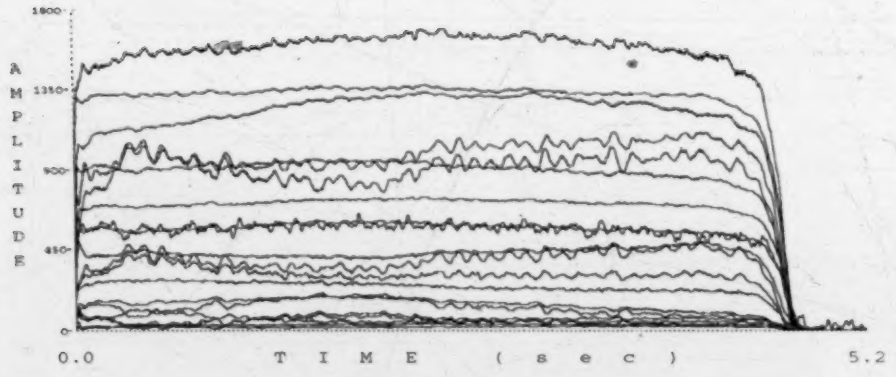


(b) PLA using absolute Euclidean error.

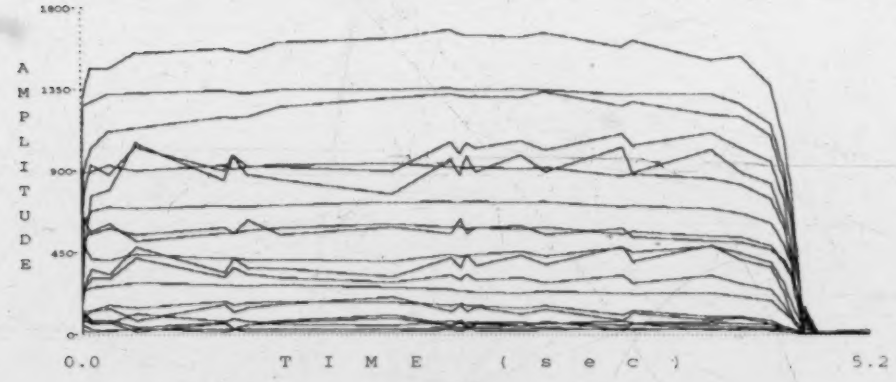


(c) PLA using relative Euclidean error.

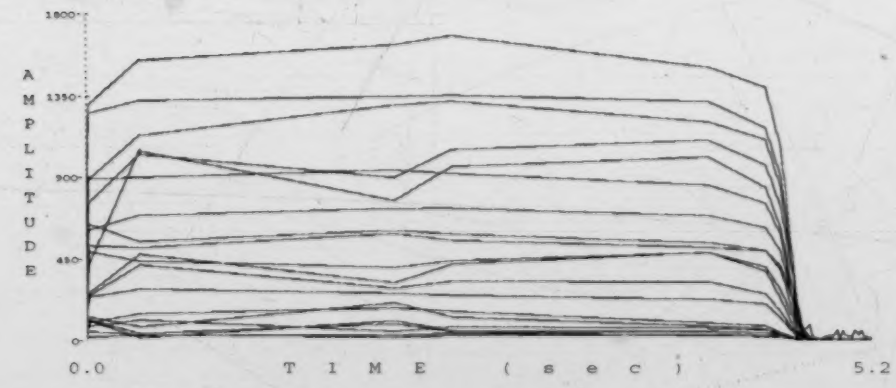
Figure 4.3: Relative vs. absolute error in PLA's of a violin tone at pitch A4.



(a) Phase vocoder analysis of the original tone.



(b) PLA using absolute Euclidean error.



(c) PLA using relative Euclidean error.

Figure 4.4: Relative vs. absolute error in 24-breakpoint PLA's of a bassoon tone at pitch C#2.

specified for the attack segment as a whole; the weight applied to the error of the rest of the tone would then be $w_2 = 1 - w_1$. Horner and Beauchamp [43] used this approach, specifying an average weighted relative error measure as

$$\frac{w_1}{F_{\text{peak}} - 1} \sum_{n=1}^{F_{\text{peak}}-1} \sqrt{\frac{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{\text{har}}} a_k(n)^2}} + \frac{w_2}{N_{\text{frames}} - F_{\text{peak}} + 1} \sum_{n=F_{\text{peak}}}^{N_{\text{frames}}} \sqrt{\frac{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{\text{har}}} a_k(n)^2}} \quad (4.4)$$

where F_{peak} is the frame number in which the peak RMS amplitude within the first 100 ms occurs; they used $w_1 = w_2 = 0.5$.

In the second option, a weight w may be specified which is to be applied to the error of each frame in the attack segment; all frames in the rest of the tone are assumed to have a weight of 1. Thus, total frame-weighted (non-relative) Euclidean error is specified as

$$\sum_{n=1}^{N_{\text{attack}}} w \sqrt{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2} + \sum_{n=N_{\text{attack}}+1}^{N_{\text{frames}}} \sqrt{\sum_{k=1}^{N_{\text{har}}} (a_k(n) - a'_k(n))^2} \quad (4.5)$$

where N_{attack} is the number of frames in the attack portion of the tone, which is calculated from a user-specified attack duration.

This approach has the advantage of applying a consistent weighting to approximations in the attack portion, regardless of the length of the tone. For example, all frames in the attack portion might be given a weight two or three times that of the frames in the remainder of the tone. By comparison, the weighting applied to each frame of the attack segment by Horner and Beauchamp's segment-weighted error measure is dependent on the relative length of the attack portion. In the segment-weighted scheme, assuming that $w_1 = (1 - w_2)$ and that frames in the sustain and release portions of the tone are unweighted (or weighted at 1), each frame in the attack portion is weighted by

$$\frac{w_1 \cdot (N_{\text{frames}} - N_{\text{attack}})}{N_{\text{attack}} \cdot (1 - w_1)}. \quad (4.6)$$

If $w_1 = w_2 = 0.5$, this reduces to

$$\frac{N_{\text{frames}} - N_{\text{attack}}}{N_{\text{attack}}} \quad (4.7)$$

so the weight of attack frames is inversely proportional to the length of the attack portion relative to the length of the rest of the tone. For example, if the attack portion is 10% of the length of the whole tone (e.g., 100 ms of a 1-second tone) and $w_1 = 0.5$, each frame of the attack is weighted at 9, but

if the attack is 2% of the tone length (e.g., 100 ms of a 5-second tone), each attack frame is weighted by 49.

Based on informal visual inspection of the amplitude-segment approximations produced by the segment-merging algorithm, it was judged that an attack bias was needed only to capture the chiff[3] at the start of several flute tones. Attack frame weightings of 4 to 12 were applied over attack durations of from 50 ms to 165 ms of tones varying in length from 3.4 to 4.4 seconds. Figure 4.5 shows the first 100 ms of a flute A#4 tone, illustrating in part (a) the first 20 harmonics of the phase vocoder analysis of the chiff which precedes the tone proper; in (b), the PLA of the first 100 ms as produced by the segment-merging algorithm without attack bias; and in (c), a PLA of the chiff, achieved with a 12-times weighting of frames in the first 50 ms of the tone.

4.2.3 Number of Breakpoints

Informal visual inspection was also used to determine how many breakpoints or what approximation error level to use as the target for the breakpoint selection algorithm. The default was to use 24 breakpoints, which is about twice the number that were found necessary in formal listening tests for the average subject to misidentify synthetic tones as real about 50% of the time [43]. However, the number of breakpoints used varied with the nature and complexity of the tone being modeled. Double the default number were used for the English horn and oboe tones and as the minimum number of breakpoints (along with an upper bound on error) for the flute and string tones; the increased number of breakpoints were required due to the low-amplitude vibrato in the double-reed instruments, as illustrated in Figure 4.6 and the high-amplitude vibrato in the flute and strings, as seen in Figure 4.2 above.

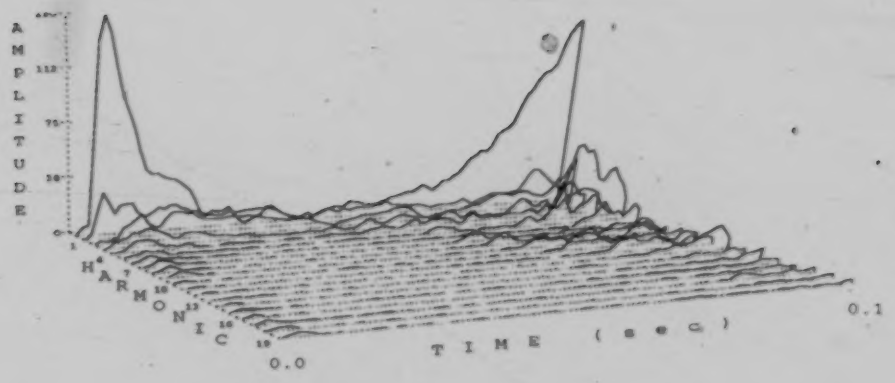
4.3 Wavetable Bank Selection

4.3.1 Use of a Clustering Algorithm

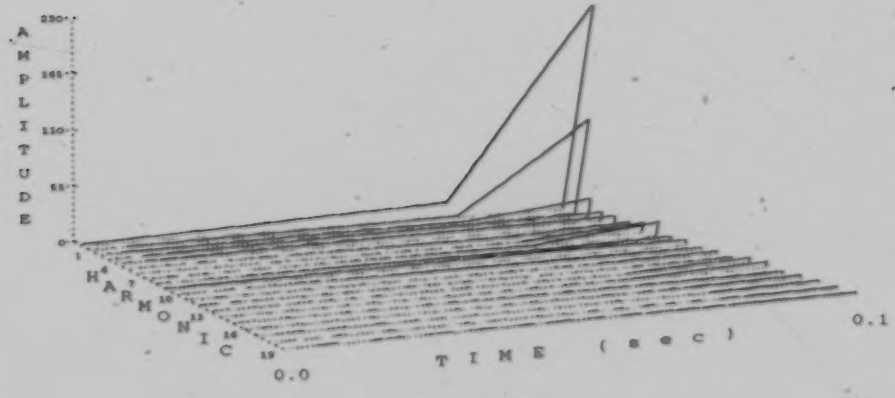
The decision to use a clustering algorithm as the basis of the method for selecting basis spectra for the wavetable bank for this research was based on the special characteristics of the problem chosen for study, which requires that the basis spectra be generally useful over a broad range of instrumental timbres and pitches.

Horner's research on the use of principal components analysis (PCA) and a genetic algorithm (GA) for wavetable selection [41, 44] revealed that the GA was the preferred method, especially for matches using fewer than four wavetables. However, the fitness function of the GA must evaluate (or at

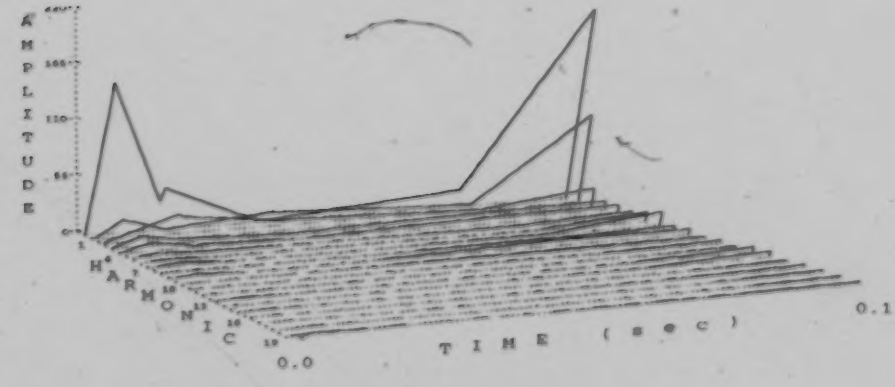
[3] *Chiff* is the sound of an initial burst of air prior to the start of phonation due to the vibration of a column of air, commonly heard at the start of flute, pan pipe, and pipe organ tones.



(a) Phase vocoder analysis of a flute chiff.

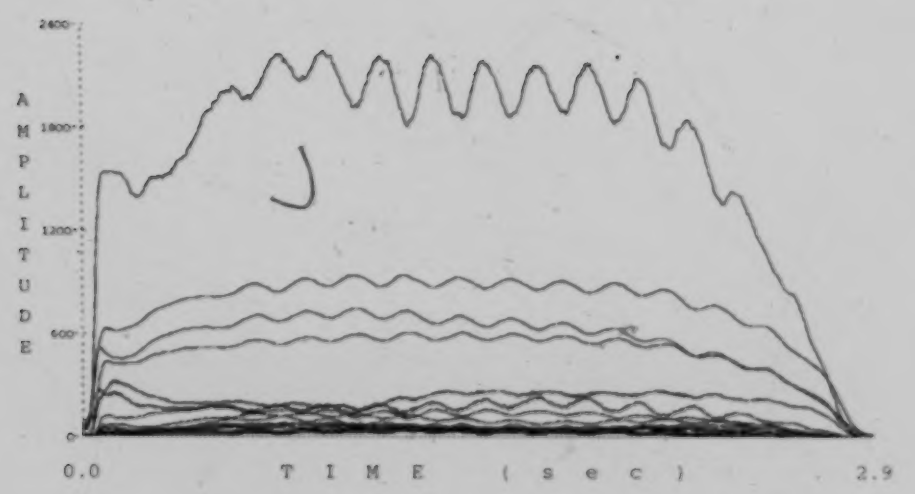


(b) PLA without attack weighting.

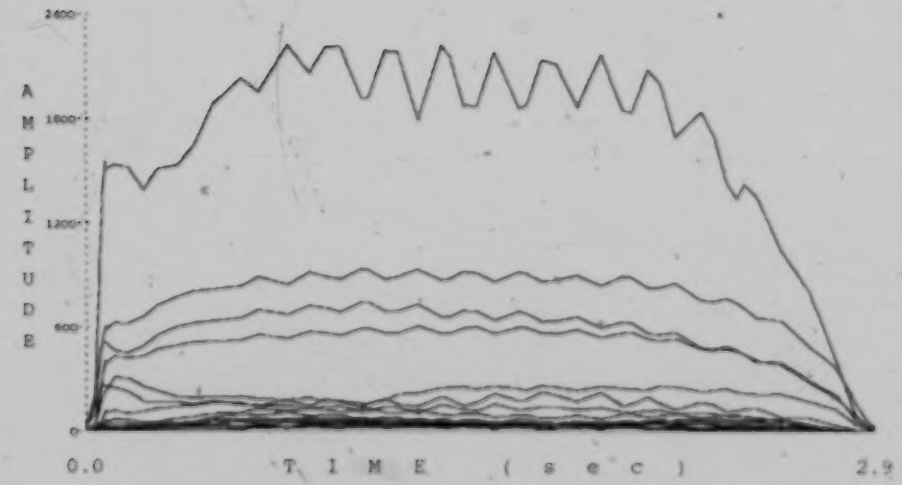


(c) PLA with attack weighting.

Figure 4.5: Piecewise linear approximation of the first 100 ms of a flute playing pitch A#4, showing the initial chiff and the start of phonation.



(a) Phase vocoder analysis (first 20 harmonics).



(b) A 48-breakpoint PLA of the tone.

Figure 4.6: Low-amplitude vibrato in an English horn tone at pitch G3.

least estimate) how well the wavetables selected in the current generation will serve as basis spectra in the matching phase. Since the problem on which this research is focused requires the selection of the best subset of the basis spectra and evaluation of each selected subset as a weighted match at each breakpoint of each instrumental tone at each pitch in the current group, the computational cost of the fitness function would be prohibitive.

Sandell and Martens [83] addressed the problem that if PCA is simply applied to the spectra of a tone, it emphasizes the louder spectra of the sustain portion of the tone at the expense of the quieter spectra of the attack and release portions; their method maps the analysis frames of a tone of any duration onto a common-resolution (200-partition) non-linear time domain which dedicates about 40% of the partitions to the attack portion of the tone. It is possible that this method could be extended for use across many tones from many instruments—Sandell and Martens reported testing tones at one pitch on several different horns—but the statistical nature of PCA suggests that more basis spectra would be needed to match a particular tone if the basis spectra were more general in nature.

The use of a clustering algorithm on the breakpoint spectra selected in the previous step was expected to find the similarities between spectra from different sources and different portions of their originating tones, yet to preserve the diversity of the various clusters of spectra. As will be shown in §5.4, these expectations were fulfilled: while many clusters consisted primarily or exclusively of adjacent spectra from the sustain portion of a single tone or from nearby tones from the same instrument, other clusters grouped spectra from the attack portions of a variety of tones and instruments, some captured similarities between attack spectra and release spectra, and some gathered similar spectra from different portions of different instrumental tones. The spectra of a given tone were typically distributed between four to seven different clusters, with most of the spectra in one or two clusters and the remaining spectra scattered across two to six others.

4.3.2 Grouping of Sample Tones by Pitch

Breakpoint spectra were normalized prior to clustering, such that the sum of the harmonic amplitudes was equal to one for each spectrum. The spectra were divided into groups, prior to clustering, according to the pitch of the tone from which they originated, and the number of harmonics to be used as attributes by the clustering algorithm (and which would be retained in the wavetable bank for matching and subsequent synthesis) was limited by the relationship of the Nyquist frequency of the sample rate to be used for synthesis and the highest anticipated fundamental frequency to be generated by pitch-shifting synthesis.

For example, for testing purposes, 198 sample tones played by sixteen dif-

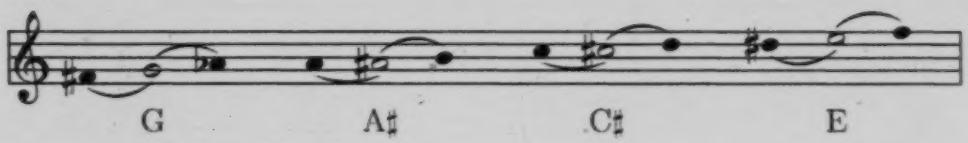


Figure 4.7: Pitch classes selected for testing (shown as open notes) and the pitches they could be shifted to synthesize (shown as solid notes).

ferent instruments[4] were selected from the *McGill University Master Samples* collection to span the range from A1 to B6, a range which includes the playable ranges of most orchestral instruments.[5] Tones were selected a minor third (three semitones) apart—all tones of pitch classes A♯ (B♭), C♯ (D♭), E and G in the specified range—so that all other tones in the range could be synthesized by pitch-shifting the control data a single semitone upward or downward; for example, the control data for G4 could also be used to synthesize F♯4 and G♯4, that for A♯4 could also synthesize A4 and B4, and so on, covering all twelve semitones of the octave, as illustrated in Figure 4.7. Since all the breakpoint spectra of the selected tones at these four pitches—G4, A♯4, C♯5, and E5—were grouped together (in what will be referred to as Group 3 in Chapter 5), they will all be resynthesized using wavetables from the wavetable bank for Group 3; furthermore, any other tones in the range from F♯4 (the first solid note in Figure 4.7) to F5 (the last solid note in the figure) will be synthesized by pitch-shifting using wavetables from the same wavetable bank. The nominal fundamental frequency of the highest note in this range (F5) is about 698 Hz (assuming the use of the international standard tuning of A440); if the tone to be synthesized at F5 is to include a vibrato of up to 2% variation in frequency, a maximum fundamental frequency of 712 Hz might be reached. If the synthesis is to be performed at CD quality—a sampling rate of 44100 Hz—then the Nyquist frequency of 22050 Hz will require that the wavetables in the wavetable bank for this group be restricted to no more than $22050/712 \approx 31$ harmonics. Similarly, since the highest pitch that would be synthesized from Bank 2 would be F4, with a nominal fundamental frequency of about 349 Hz and a highest expected fundamental frequency due to vibrato of about 356 Hz, wavetables in Bank 2 should be restricted to 61 or 62 partials for CD-quality synthesis.

[4] Because the *McGill University Master Samples* collection does not include tones spanning the full range of each member of the saxophone family (bass, baritone, tenor, alto, and soprano) but uses about an octave from each instrument such that the recorded tones span the full range of the family, the saxophones were regarded as a single instrument for the purposes of this study.

[5] This range excludes the lowest and highest octaves of the piano (the range of which extends downward to A0 and upward to C8), the lowest string of the double bass (which, with an extension, can descend as low as C1), the bottom octave playable by a contrabassoon, the upper octave of the glockenspiel and xylophone, and the top fifth of the piccolo's range.

4.3.3 Selection of Class Representatives

The publicly available[6] unsupervised Bayesian classification[7] program `AutoClass C` [14] was used to perform clustering of all the breakpoint spectra from each selected instrumental tone in each group. Given a database of attribute vectors and a class model, `AutoClass` finds the set of clusters that is maximally probable with respect to the data and model. The number of clusters is determined automatically. The "single normal CN" model was used for each attribute; this models each real-valued attribute (normalized harmonic amplitude) with a conditionally independent Gaussian normal distribution, assuming that there are no missing values and that the measurement error is both constant and small relative to the model variance.

The breakpoint spectrum which was nearest (in a Euclidean sense) to the centroid of the spectra which were clustered together was then selected as the representative of that cluster (class) and, as such, was provisionally selected as one of the wavetables in the wavetable bank for that group.

Wavetable matching was then performed using this provisional wavetable bank for all the tones in the current group, and two types of statistics were gathered: the matching error and the usage of the wavetables in the bank. On the basis of this information, the selection of spectra in the wavetable bank was then hand-tuned in one or both of two ways:

- If one or more tones in the group had a particularly high matching error, the breakpoints of the tone(s) which had higher-than-average matching error were examined to see if the addition of one of the high-error breakpoint spectra to the wavetable bank would be likely to reduce the matching error at a number of breakpoints.

- If one of the wavetables in the bank was used significantly fewer times in wavetable matches than the other wavetables in the bank, that wavetable was removed from the bank, since a secondary objective of the wavetable bank selection process was to find as small a set of wavetables as possible which would still support low-error matches to the breakpoints of the group.

After the wavetable banks were improved by such modifications, if applicable, the breakpoint matching procedure was repeated to verify that the modifications had been effective.

---

[6] Available online at `http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass/autoclass-c-program.html` (2002-10-01).

[7] Although the authors of `AutoClass` refer to it as a classification program, it would more typically be called a clustering program, since it groups data points without reference to a priori category labels. According to common usage, a classifier assigns labels to new data points given a set of labelled data points as a training set. In the following, the process of grouping spectra based on their similarity will be referred to as clustering, but the set of spectra which have been grouped together as a result of clustering will be referred to interchangeably as a cluster or a class, since each spectrum could be labelled subsequent to clustering and new spectra could be classified relative to the existing groups.

4.4 Breakpoint Matching

4.4.1 Implementation of Multi-Level Search

As was explained in §3.2.2, if the size of the initial match to each breakpoint spectrum were less than the number of oscillators to be used on resynthesis, a second-level search could be used to augment the initial matches. This second-level search is in effect a pruned search, since it seeks to augment only those sets of wavetables which were actually used as initial matches by the first-level search (whether it was an exhaustive search or a GA). However, it is a significant implementation detail that the second-level search tests the augmented wavetable sets not only for improved matches at those breakpoints which used the original (unaugmented) wavetable set as a best match in the first-level search, but at all breakpoints; quite often, an augmented wavetable set was found to be the best match at breakpoints which, after the first-level search, were previously best matched by entirely different wavetable sets.

For example, Figure 4.8 compares the two-wavetable matches to the breakpoints of a French horn E2 tone found by a first-level exhaustive search with the three-wavetable augmented matches found by a second-level search. (Part (b) of this figure is the same as part (a) of Figure 3.13.) The three-wavetable sets at breakpoints 2, 9, 10, 15, and 18 are highlighted in part (b) of the figure to indicate that these sets are not supersets of the two-wavetable matches found by the first-level search at these breakpoints. Although every possible superset of the set {15, 38} would have been tried at breakpoint 2 by the second-level search, it turned out that a superset of the set {6, 15} which was initially used only at breakpoint 3 also happened to be the best three-wavetable match to breakpoint 2. It also happened that various supersets of the set {4, 15} which was initially used at breakpoints 4 and 19–22 proved to be the best matches to breakpoints 4, 5, 15, and 18–22; at breakpoints 15 and 18, the set {4, 15, 23} was found to be a better match than any supersets of the two-wavetable matches previously found to be best at those breakpoints. Of particular interest is the replacement of the set {19, 30} by {10, 15, 37} at breakpoints 9 and 10 because this results in an eight-breakpoint run of the latter set from breakpoint 7 to 14; as shown in Figure 3.14, this wavetable set and four-table supersets of it were finally selected for use by the oscillator-assignment optimizer at all the breakpoints of this tone from 1 to 19.

To avoid redundant second-level searches, all the sets of wavetables found to be best matches at one or more breakpoints by the first-level search were inserted in a set of wavetable sets prior to performing the second-level search *per se*. This also ensured that a second-level search was conducted for every set which was found to be a best match by the first-level search. For example, all supersets of {19, 30} would have been tested at all the breakpoints of the horn E2 tone of Figure 4.8, even though that set would have been replaced by the match {10, 15, 37} at breakpoints 9 and 10 prior to the initiation of the search to augment {19, 30}; it may well have been (but was not in this

1:	10	29	
2:	15	38	
3:	6	15	
4:	4	15	
5:	15	47	
6:	15	37	
7:	15	37	
8:	15	37	
9:	19	30	
10:	19	30	
11:	15	37	
12:	15	37	
13:	15	37	
14:	15	37	
15:	15	37	
16:	15	37	
17:	15	37	
18:	15	46	
19:	4	15	
20:	4	15	
21:	4	15	
22:	4	15	
23:	10	30	
24:	16	46	

(a) Initial 2-wavetable matches.

1:	10	26	29
2:	**6**	**9**	**15**
3:	6	9	15
4:	4	15	21
5:	4	15	47
6:	6	15	37
7:	10	15	37
8:	10	15	37
9:	**10**	**15**	**37**
10:	**10**	**15**	**37**
11:	10	15	37
12:	10	15	37
13:	10	15	37
14:	10	15	37
15:	**4**	**15**	**23**
16:	10	15	37
17:	6	15	37
18:	**4**	**15**	**23**
19:	4	15	23
20:	4	15	27
21:	4	10	15
22:	4	15	37
23:	9	10	30
24:	16	45	46

(b) Augmented "2+1" matches.

Figure 4.8: Initial and augmented matches to the breakpoints of a French horn playing pitch E2.

case) that some superset of {19, 30} would have been still better as a match at some breakpoint than {10, 15, 37}.

In order to further prune the second-level search, the use of an error threshold above which augmenting should be performed was tested; if the match found by the first-level search for a particular breakpoint had an error less than the specified threshold, the second-level augmenting search would be skipped for that wavetable set. However, it was found that this attempt to reduce search time resulted in poorer matches on average, since it was quite often the case that a wavetable set which was a good match to some breakpoint spectrum proved to be a good match at other breakpoints as well when augmented.

4.4.2 Implementation of the Genetic Algorithm

The genetic algorithm which was used as an alternative method of finding initial matches to breakpoint spectra was implemented using Matthew Wall's *GAlib: A C++ Genetic Algorithm Library*.[8] The n-table match at each breakpoint of a tone was represented as a two-dimensional array of integers using a customization of *GAlib*'s `GA2DArrayAlleleGenome` template type; the allele set (the set of values that a gene may assume) was the integers in the range $1 \ldots N_{\text{tables}}$, where N_{tables} is the number of wavetables in the bank for the relevant group of tones. The GA was run with a population size of 100, a crossover probability of 0.9, a mutation probability of 0.1 (overall, not per gene), linear scaling, stochastic remainder sampling selection, and termination upon convergence of 99% over 50 generations (i.e., when the best score of the population 50 generations ago divided by the best score of the current population is at least 0.99).

As mentioned in §3.2.2 (page 52), the GA was not competitive with exhaustive search either in terms of time or in terms of matching error unless caching of intermediate results was used. When the objective function was invoked by the GA on an individual of the current generation, the genes of that individual were deemed to be an ordered list of wavetable sets, each to be used as an initial match at its respective breakpoint, and were inserted into a multimap which mapped each wavetable set to the breakpoint(s) at which it was used. The multimap was then traversed with an iterator so that the basis spectra corresponding to each different wavetable set used in the current individual were referenced as a matrix and analyzed by LUP decomposition as discussed in §3.2.1; the LUP decomposition was then used as in equation 3.9 to fit that wavetable set in a least-squares sense to the spectrum at each breakpoint specified by the genes of the current individual, again using the multimap iterator. Without caching, this implied that LUP decompositions were being repeatedly performed on the same sets of wavetable spectra, since it was highly likely that

[8] Available online at `ftp://lancet.mit.edu/pub/ga/`, with documentation available at `http://lancet.mit.edu/ga/` (2002-10-01).

the same wavetable sets would be used by multiple individuals in the current generation and repeatedly from generation to generation; similarly, the same wavetable sets would likely be used as matches to the same breakpoint spectra in various individuals and across generations. For example, a three-table GA match to a 24-breakpoint PLA of a bassoon A♯1 tone using the wavetable bank for group 1 (which has 48 wavetables, each consisting of 146 harmonic amplitudes), a population size of 50, and a number of generations to convergence of 25 converged after 247 generations, during which it performed 269,632 LUP decompositions and 269,688 least-squares solutions and error calculations in about 146 seconds on the test platform; augmenting the matches with a fourth wavetable and optimizing the final oscillator allocation took an additional 35 and 34 seconds, respectively, for a total time of 215 seconds. By comparison, an exhaustive search of depth 3 performed $\binom{48}{3} = \frac{48!}{3!(48-3)!} = 17{,}296$ LUP decompositions and 24 times that many (415,104) least-squares and error evaluations in just over 110 seconds; augmentation required only 6.5 seconds and optimization, 22 seconds, for a total of about 139 seconds.

Caching was implemented by introducing two mappings as static data members of the objective function so that the contents of the mappings would be preserved across the evaluations of all individuals in all generations. The first maps from sets of wavetables to the results of LUP decompositions, the second, from breakpoint number and wavetable set to the corresponding least-squares solution. When iterating across the multimap, the first map is checked for each wavetable set in the current individual, and LUP decompositions are performed only for those wavetable sets not already in the map; similarly, when iterating across the breakpoints at which a given wavetable set is used in the current individual, the second map is searched for a pre-existing least-squares solution and error level. As a result of this form of caching, a GA search for a three-table match to the same bassoon tone discussed above now performs only 7,331 LUP decompositions and 9,248 least-squares and error evaluations in just over 16 seconds, for a total time of 85 seconds. While this particular invocation of the GA[9] tried only about 42% of the possible three-wavetable combinations, each at an average of 1.26 breakpoints, the final result (after augmentation and optimization) had an average error rate only two-thirds of a percent worse than that found by a "3+1" exhaustive search.

However, the primary and secondary contributors to the low error rate of this matching were the second-level exhaustive search to augment the initial three-table matches and the subsequent overlapping of wavetable sets and optimization of oscillator assignments using the shortest path algorithm. The average squared error across the 24 breakpoints of the PLA to the bassoon A#1 tone of the three-wavetable matches found by the GA was 24 times the average squared error of the best matches found by a first-level exhaustive search; surprisingly (and atypically), the average squared error of the four-

[9]Since the GA is probabilistic, invocations with other random seeds would find different results.

table matches found by augmenting the initial matches found by the GA was two percent *lower* than the error after augmenting the matches found by exhaustive search. This result is understandable when one considers that the augmenting search tries augmenting each wavetable set found by the initial search with every other wavetable in the bank and tests each possible augmentation at every breakpoint; since the GA match had more diversity in its initial matches, the augmenting search had over five times as many possibilities to try than with the initial matches found by exhaustive search, and it happened to find one or more fortuitous combinations of wavetables. This greater initial diversity also resulted in larger average sizes of the wavetable sets constructed by overlapping—11.1 compared to 10.2—but did not continue to contribute to a favourable final outcome: the error of the globally optimized result found by the shortest path algorithm was slightly higher[10] for the method which began with a GA search; nonetheless, this result is remarkable considering that the method using the GA took only 61% of the time of the method beginning with exhaustive search.

If the goal of applying this method of analysis is not to find the best overall matches at any cost, but rather to find good matches fairly quickly, then the method using a GA with a moderate population size (about 50) and early termination (about 25 generations to convergence) is recommended. For this research, it was decided to use the more conservative GA parameters—a population size of 100 with 50 generations to convergence—since testing showed that a "3+1" search using a GA with these parameters to find the initial three-table matches yielded a better final result on average than a GA with the less conservative parameters;[11] however, these parameters make the GA method about 10% slower than the method using exhaustive search for "3+1" searches, while the GA with less conservative parameters runs in about half the time of a "3+1" exhaustive search. Furthermore, it is not recommended that a GA be used alone to construct the initial matches which, after overlapping, will be used as input to the shortest path algorithm: a "4+0" search using a GA with conservative parameters resulted in final matches with an error level 9% higher than those found by a "3+1" two-level exhaustive search, and took 1.8 times as long; a "4+0" search using a GA with a smaller population and early termination took about the same time as a "3+1" exhaustive search, but yielded error levels over 30% higher.

4.4.3 Implementation of Overlapping

As discussed in §3.2.3, a third-level exhaustive search is performed to re-augment any wavetable sets which, after overlapping has been performed, are

[10] About two-thirds of a percent higher.

[11] Specifically, the GA with conservative parameters yielded matches with error levels within 1.7% of both the "4+0" and "3+1" matches found for tones in Group 1 with an initial exhaustive search, compared with an increased error of 4.0% on average by using the less conservative GA parameters.

still smaller in size than the number of oscillators to be used on resynthesis. As with second-level augmentation (see §4.4.1), the set of all sets of wavetables which need re-augmentation is identified before any third-level search is performed.

The size of the graph which will be searched in the optimization phase is determined, in part, by the sizes of the wavetable sets constructed for each breakpoint by overlapping. In order to reduce the amount of work done in constructing the graph and running the shortest path algorithm, some experimentation was done on allowing the imposition of a limit on size of the wavetable sets resulting from overlapping. Overlapping at distance one—the construction of an initial wavetable set at a given breakpoint as the union of the matches at that breakpoint and the immediately adjacent breakpoint(s)—is always performed, but overlapping at greater distances are performed in order—first at distance two, then at distance three, and so on—and will be terminated for a given breakpoint if the size of the wavetable set that would have been constructed by the next level of overlapping would have exceeded the specified limit. Both this option and the option of limiting overlapping to a distance of one proved to be useful for five-oscillator optimizations, which otherwise would have required significantly greater computation time and memory.

4.4.4 Implementation of the Optimizer

The construction of the DAG on which the single-source acyclic weighted shortest path algorithm will be invoked must take into account the requirement to fade in a wavetable which will begin to be used at some internal breakpoint and to fade out one which ceases to be used. The construction of sets of wavetables eligible for use at each breakpoint through overlapping allows the optimizer to schedule a wavetable for use at breakpoints preceding and/or following the breakpoint or sequence of breakpoints at which it is a member of the best match. The algorithm which adds vertices to the DAG must also generate vertices which do *not* include all the wavetables of the best match at a given breakpoint so that one or more oscillators can be used to fade out a wavetable from the previous breakpoint to the current one and to fade in another wavetable from the current breakpoint to the next.

This implies, in general, that if the number of available oscillators is N_{osc} and the number of wavetables in the wavetable set for the current breakpoint is N_{wt}, there will be at most

$$\binom{N_{wt}}{N_{osc}} + \binom{N_{wt}}{N_{osc}-1} + \cdots + \binom{N_{wt}}{1} \tag{4.8}$$

vertices associated with each breakpoint. For example, if $N_{osc} = 3$ and the current wavetable set is {2, 3, 5, 8}, vertices could be added to the graph to represent the following wavetable selections at the current breakpoint:

{2, 3, 5}, {2, 3, 8}, {2, 5, 8}, {3, 5, 8}

{2, 3}, {2, 5}, {2, 8}, {3, 5}, {3, 8}, {5, 8}

{2}, {3}, {5}, {8}

This number may be reduced by two constraints:

- Because a one-wavetable match to a breakpoint spectrum is not likely to be very accurate (except at those breakpoints for which the spectrum was selected for inclusion in the wavetable bank), provision was made for specifying a minimum number of wavetables to be used in the final matches. By default, if N_{osc} is 3, 4, or 5, the minimum number of wavetables which may be used at each breakpoint is $N_{osc}-2$; for $N_{osc} \geq 6$, the default minimum is $N_{osc}-3$. An absolute minimum of one wavetable is assumed.

- A vertex should *not* be added to the graph at the current breakpoint if there can be no outgoing edges from it to any vertex associated with the next breakpoint. To continue the example above, if the wavetable set for the next breakpoint were {2, 4, 5, 6, 8, 9}, then the algorithm should avoid generating a vertex at the next breakpoint for the wavetable selection {4, 6, 9}, since all three wavetables would have to be faded in simultaneously.

In combination, these two constraints can significantly reduce the size of the oscillator assignment graph and the amount of work done to generate it. For example, if the minimum number of wavetables allowed in a match is two, then vertices representing the matches {4, 6}, {4, 9}, {6, 9}, {2, 4, 6}, {2, 4, 9}, {2, 6, 9}, {4, 5, 6}, {4, 5, 9}, {5, 6, 9} would also be unreachable at the next breakpoint, since they would require two oscillators to have been unassigned at the previous breakpoint in order to fade in the two new wavetables.

The algorithm for generating the vertices and edges of the graph representing all possible oscillator assignments takes into account the preceding constraints and the following possibilities:

- If the set of wavetables at the current vertex of the graph is $S_{current}$ and the set of wavetables eligible for use at the next breakpoint is S_{next}, then vertices should be generated at the next breakpoint to represent the wavetable set $S_{current} \cap S_{next}$ and all the subsets of that set which are of at least the specified minimum match size.

- If $|S_{current}| < N_{osc}$ (i.e., if the match represented by the current vertex includes one or more unallocated oscillators or wavetables which have just been faded out to zero amplitude), then any of these "zero wavetables" can be replaced by any member of the set S_{next} at the next breakpoint.

These two possibilities can be used in combination as well: any or all members of the current set can become zeros at the same time as any zeros in the current set become members of the next set. (Of course, the generation of

duplicate wavetable sets at the next breakpoint is to be avoided: a zero in the current set cannot be replaced by a member of S_{next} which is already in the wavetable set of an adjacent vertex as a result of the first possibility above.)

These principles lead to the recursive algorithm presented in pseudocode in Figures 4.9–4.12. For simplicity of presentation, not all of the procedure parameters have been shown in the pseudocode; it should be assumed that the following values are available within each function:

`numOscillators`: the number of oscillators available

`minWavetables`: the minimum number of wavetables to be used in a match

`numBreakpoints`: the total number of internal breakpoints

`bestSets`: a vector, indexed by breakpoint number, of the sets of wavetables eligible for use at each breakpoint

`sink`: the final vertex of the DAG[12]

`g`: the graph to which vertices and edges are added by the algorithm

The pseudocode is Pascal-like,[13] but with the addition of object-oriented constructs such as `g.addEdge`, which should be presumed to invoke the method `addEdge` on the object `g`. The class `WavetableSelection` encapsulates a breakpoint number and a set of wavetables; for simplicity, these are accessed as the object attributes[14] `source.breakpoint` and `source.tables`, the latter corresponding to S_{current} in the discussion above. Similarly, S_{next} corresponds to `bestSets[ source.breakpoint + 1 ]`. The code also assumes the existence of a `Set` type[15] which not only supports the standard set operators such as intersection ($\cap$) and set difference ($-$) but also overloads the operator '$-$' to indicate exclusion of an individual element from a set and supports operator '+' to include an element in a set. The pseudocode uses the construct **for** ... **in** to iterate across the elements of a set, with the presumption that a copy of the set is used to perform the iteration, allowing the set to be modified in the body of the loop without disrupting the iteration.[16]

This set of interrelated procedures is invoked by the main program, which initializes the graph by adding vertices representing the two external breakpoints (at breakpoints 0 and `numBreakpoints` + 1, both with null wavetable

[12] In the actual implementation, the sink vertex is not actually passed as a parameter, but is easily constructed as `WavetableSelection( numBreakpoints + 1)`, the unique vertex with the breakpoint number of the final external breakpoint and a null set of wavetables.

[13] The pseudocode differs from Pascal in identifying the range of a **for** loop or the **then** or **else** clauses of an **if** by indentation rather than by **begin-end** pairs, and by using semicolons as statement delimiters rather than separators.

[14] They are actually implemented in C++ as accessor functions.

[15] The set type is implemented as a C++ templated type.

[16] This is actually implemented in C++ using an operator that retrieves an element of the set and iteratively removes each retrieved element from the set until the set is empty.

```
procedure GenerateEdges ( source : WavetableSelection )
  var
    currentSet, nextSet, commonSet : Set of WavetableType;
    numZeros : integer;
  begin
    Base case for recursion
    if source.breakpoint = numBreakpoints then
      g.addEdge( source, sink );
    else
      currentSet ← source.tables;
      nextSet ← bestSets[ source.breakpoint + 1 ];
      commonSet ← currentSet ∩ nextSet;

      Add vertices to the graph for the set of wavetables common to the set
      at the current vertex and the set of eligible wavetables for the next
      breakpoint and for all subsets of that set which are at least as large as
      the minimum match size, and add edges from the current vertex to each
      of the new ones.
      if |commonSet| ≥ minWavetables then
        AddVertexAndEdge( source, commonSet );
        AddSubsets( source, commonSet, commonSet );

      Add wavetables from the next set of eligible wavetables to replace zeros
      (unallocated oscillators) in the current set of wavetables.
      numZeros ← numOscillators - |currentSet|;
      if numZeros > 0 then
        ReplaceZeros( numZeros, source, commonSet, commonSet,
                      nextSet - commonSet );
  end GenerateEdges
```

Figure 4.9: Procedure `GenerateEdges`.

```
procedure AddSubsets ( source : WavetableSelection;
                       wavetableSet : Set of WavetableType;
                       removableSet : Set of WavetableType )
  var
    element : WavetableType;
    subset : Set of WavetableType;
  begin
    For each wavetable in the current set, add a vertex for the next breakpoint
    that substitutes a zero for that wavetable. If the current set has had wave-
    tables added to replace zeros, remove only those elements which were in
    the original current set (the 'removableSet').
    for element in removableSet do
      subset ← wavetableSet − element;
      if |subSet| ≥ minWavetables then
        AddVertexAndEdge( source, subset );

      Recur if more than one wavetable can be faded out at once.
      if |subSet| > minWavetables then
        removableSet ← removableSet − element;
        AddSubsets( source, subset, removableSet );
  end AddSubsets
```

Figure 4.10: Procedure `AddSubsets`.

```
procedure ReplaceZeros ( numZeros : integer;
                         source : WavetableSelection;
                         currentSet, superset, nextSet :
                             Set of WavetableType )
  var
    element : WavetableType;
    augmentedSet : Set of WavetableType;
  begin
    Add an edge from the current wavetable set to the same set with one or
    more zeros replaced by wavetables from the next set of eligible wavetables.
    Here, 'nextSet' contains only those elements not in 'currentSet'.
    for element in nextSet do
      augmentedSet ← superset + element;
      if |augmentedSet| ≥ minWavetables then
        AddVertexAndEdge( source, augmentedSet );

      One or more of the wavetables in the current set could be faded out
      while the new wavetables are fading in.
      if |augmentedSet| > minWavetables then
        AddSubsets( source, augmentedSet, currentSet );

      Recur to handle current sets with multiple zeros.
      if numZeros > 1 then
        nextSet ← nextSet − element;
        ReplaceZeros( numZeros − 1, source, currentSet,
                      augmentedSet, nextSet );
  end ReplaceZeros
```

Figure 4.11: Procedure `ReplaceZeros`.

```
procedure AddVertexAndEdge ( source : WavetableSelection;
                             nextSet : Set of WavetableType )
  var
    destBrkpt, numZeros : integer;
    continuingSet : Set of WavetableType;
    dest : WavetableSelection;
    vertexAdded : Boolean;
  begin
    Don't generate a vertex at this breakpoint if there can be no outgoing edges
    from it to any vertex at the next breakpoint.
    destBrkpt ← source.breakpoint + 1;
    if destBrkpt < numBreakpoints then
      numZeros ← numOscillators - |nextSet|;
      continuingSet ← nextSet ∩ bestSets[ destBrkpt + 1 ];
      if |continuingSet|+ numZeros < minWavetables then
        return;

    Create a new wavetable selection and see if it is already in the graph. If
    so, just add the edge from source to destination; if not, add the edge and
    then recur to generate outgoing edges from the new vertex.
    dest ← WavetableSelection( destBrkpt, nextSet );
    vertexAdded ← g.addVertex( dest );
    g.addEdge( source, dest );
    if vertexAdded then
      GenerateEdges( dest );
  end AddVertexAndEdge
```

Figure 4.12: Procedure `AddVertexAndEdge`.

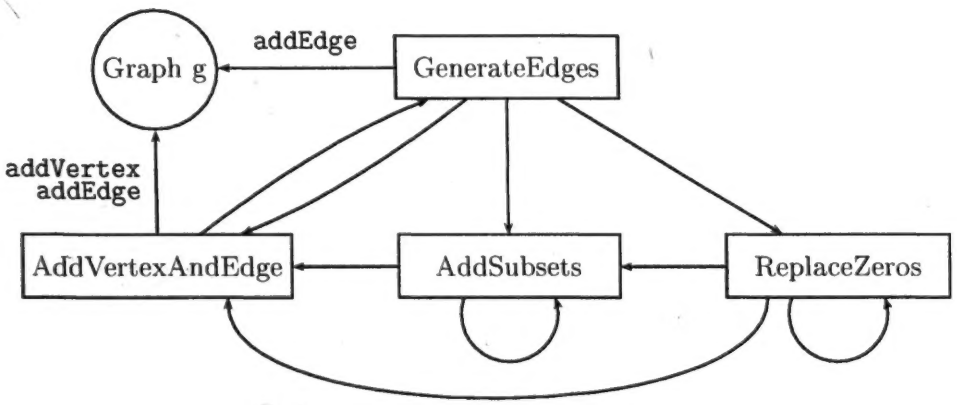


Figure 4.13: Call diagram for **GenerateEdges** and related procedures.

sets) and then calls **GenerateEdges** (Figure 4.9) with the start vertex as a parameter. The procedures are mutually recursive, since the procedure **AddVertexAndEdge** (Figure 4.12) calls **GenerateEdges** if it has added a new vertex to the graph. Figure 4.13 is a call diagram illustrating the interrelationships between **GenerateEdges** and its related procedures.

Procedure **AddSubsets** (Figure 4.10) recursively generates the subsets of $S_{\text{current}} \cap S_{\text{next}}$ which are at least of the minimum specified match size and calls **AddVertexAndEdge** to add vertices representing these subsets to the graph, along with edges from S_{current} to the new vertices. Each edge from S_{current} to a proper subset of S_{current} represents the fade-out of one or more wavetables from one breakpoint to the next.

Procedure **ReplaceZeros** (Figure 4.11) recursively generates supersets of S_{current} if it is smaller in size than N_{osc}, iteratively replacing the "zero" wavetables in S_{current} with each of the elements in $S_{\text{next}} - S_{\text{current}}$ in turn. This procedure also implements the possibility that one or more wavetables may be faded in at the same time as one or more other wavetables are faded out by calling **AddSubsets** with each superset that it generates; however, it uses the **removableSet** parameter of **AddSubsets** to specify that subsets may be generated only by removing wavetables that were in the current wavetable set originally, not by removing those that were just added to replace "zeros."

Procedure **AddVertexAndEdge** (Figure 4.12) is called by each of the previous three procedures to add new vertices and edges to the graph. The graph is implemented as a vector of vertices, where each vertex contains an adjacency list of edges; this structure is supplemented by a mapping from vertex names (labels) to the indices of the respective vertices in the vector.[17] The graph **addVertex** method first tries to add a vertex label (a **WavetableSelection** consisting of the next breakpoint number and the set of wavetables **nextSet**)

[17]This graph implementation is a variation on the data structure presented by Mark Allen Weiss [94].

to the mapping, receiving in return a value indicating whether this key was new or was found in the mapping;[18] if the key was new, it then creates a new vertex and labels it with the specified name. The `addEdge` method is then called to add an edge to the graph from the `source` parameter of `AddVertexAndEdge` to the destination vertex (whether new or pre-existing). If the vertex was newly added to the graph, `AddVertexAndEdge` calls `GenerateEdges` to generate by mutual recursion any outgoing edges from the new vertex to vertices at the subsequent breakpoint.

Before adding a new edge to a potentially new vertex in the graph, the procedure checks whether the second constraint listed above applies: it suppresses the addition of the vertex at the destination breakpoint if it can be determined that there can be no outgoing edges from that vertex to any vertex at the subsequent breakpoint. For example, if `numOscillators` is 3, `minWavetables` is 2, the degree of overlapping is 1, and the best matches at breakpoints u, v, and w as found by a "2+0" search (for the sake of the simplicity of the example) are $\{1, 2\}$, $\{3, 4\}$, and $\{5, 6\}$, respectively, then there will be no outgoing edges from any vertex at breakpoint v of the form $\{1, 2\}$ or $\{1, 2, n\}$, where n is any wavetable index in the range $3 \ldots 6$; this is so because wavetables 1 and 2 are eligible for use (by virtue of overlapping at a distance of 1) at breakpoints u and v, but not at breakpoint w, and it is not possible to fade out both wavetables 1 and 2 between breakpoints v and w if at least two wavetables must be in use (with non-zero weightings) at each breakpoint. More generally, such vertices (wavetable sets) may be detected as those for which the sum of the cardinality of the intersection between the original set and the eligible set for the breakpoint after next and the number of zeros (unallocated oscillators) in the original set is less than the minimum number of wavetables to be used at each breakpoint (except at the last internal breakpoint, since all wavetables at that breakpoint will be faded out by the final, external breakpoint).

There is another class of vertices that should be suppressed because they end up having no outgoing edges to the following breakpoint because the only wavetable set to which they lead is suppressed at the breakpoint after that. To continue the example above, and further presuming that wavetables 3 and 4 are not in the best match at the breakpoint following w, then there will be four edges from vertices at breakpoint u to each of the vertices $\{1, 3, 4\}$ and $\{2, 3, 4\}$ at breakpoint v, but there will be no outgoing edges from either of these vertices, since neither wavetable 1 nor 2 belongs to the set of eligible wavetables for breakpoint w and the generation of vertex $\{3, 4\}$ was therefore suppressed at breakpoint w because both wavetables 3 and 4 must be faded out from w to the following breakpoint. However, such vertices are few in number and it would be more expensive to eliminate them than to leave them in the graph and let the shortest path algorithm discover their uselessness.

[18] The `map` associative container of the C++ Standard Template Library (STL) offers an `insert` method which returns a pair, the second element of which is a `bool` which is `true` if the operation actually inserted a new value in the mapping or `false` if the key value was found in the existing mapping.

The pseudocode for `AddVertexAndEdge` does not indicate how the weight (cost) of each vertex is set to be the least-squares error of the match that it represents to the spectrum at its respective breakpoint. In the actual implementation, a multimap is used to keep track of the breakpoint/wavetable-set combinations at the vertices of the DAG so that an LUP decomposition is done only once for each different wavetable set and the least-squares fit and error calculation is then done for each use of the set at various breakpoints.

The matching error at breakpoint i, ε_i, is calculated as

$$\varepsilon_i = \sum_{k=1}^{N_{\text{har}}} (a_k(t_i) - a'_k(t_i))^2 \tag{4.9}$$

where N_{har} is the number of harmonics retained in the spectra of the relevant wavetable bank, t_i is the time index of breakpoint i, $a_k(t)$ is the amplitude of the kth harmonic of the spectrum of the original signal at time t_i (breakpoint spectrum i), and $a'_k(t)$ is the corresponding amplitude of the spectrum calculated as the weighted combination of the wavetable bank spectra referenced by the match being evaluated.

The error of the oscillator assignment found by the shortest path algorithm can be reported in a number of ways: total squared error, mean squared error, root sum-of-squares error (the Euclidean norm or 2-norm, if the breakpoint matching errors are regarded as a vector), or root mean squared (RMS) error.

$$\begin{matrix}\text{Total} \\ \text{squared} \\ \text{error}\end{matrix} = \sum_{i=1}^{N_{\text{bkpt}}} \varepsilon_i \tag{4.10}$$

$$\begin{matrix}\text{Mean} \\ \text{squared} \\ \text{error}\end{matrix} = \frac{1}{N_{\text{bkpt}}} \sum_{i=1}^{N_{\text{bkpt}}} \varepsilon_i \tag{4.11}$$

$$\begin{matrix}\text{Root} \\ \text{sum-of-squares} \\ \text{error}\end{matrix} = \sqrt{\sum_{i=1}^{N_{\text{bkpt}}} \varepsilon_i} \tag{4.12}$$

$$\begin{matrix}\text{RMS} \\ \text{error}\end{matrix} = \sqrt{\frac{1}{N_{\text{bkpt}}} \sum_{i=1}^{N_{\text{bkpt}}} \varepsilon_i} \tag{4.13}$$

RMS error is used as the measure of overall matching error for the results reported in §5.5, primarily because it provides a compact representation of error. RMS error is also comparable across tones with varying numbers of breakpoints, since it is the square-root of the mean squared error.

Chapter 5

Experimental Results

5.1 Tones Selected for Testing

As previously discussed in §4.3.2 (p. 74), a set of tones played by sixteen different instruments, spanning the range from A1 to B6 by minor thirds (as illustrated in Figure 4.7), were selected from the *McGill University Master Samples* collection for the purpose of testing the proposed analysis/synthesis method. As indicated in Table 5.1, all tones of pitch classes[1] A♯, C♯, E and G in the chosen range as played by the bassoon (abbreviated as `bsn` in the table), B♭ clarinet (`cla`), bass clarinet (`clb`), English horn (`eng`), flute (`flt`), glockenspiel (orchestral bells, `glk`), French horn (`hrn`), oboe (`obo`), piano (`pno`), the saxophone family[2] (`sax`), C trumpet (`tpt`), trombone (`trb`), viola (`vla`), string bass (bass viol or simply "bass", `vlb`), violoncello (or 'cello, `vlc`), and violin (`vln`) were selected, a total of 198 tones. For the stringed instruments, the fingered versions of the tones were preferred to the open variants.

The tones were extracted at CD quality (a sampling rate of 44100 Hz) to WAV files[3] using Heiko Eissfeldt's `cdda2wav`[4] and edited to remove leading and trailing silence (or low-level noise) using Thomas Eschenbacher's `kwave` audio file editor.[5]

Figure 4.7 also indicates the grouping of the instrument tones according to pitch ranges in order to allow the use of more harmonics in the synthesis of lower-pitched tones than for higher tones, as discussed in §4.3.2 The number

[1] All references to pitches specify sounding pitch, not written pitch.

[2] As previously mentioned in note 4 on page 75, the selected sax tones were produced by different members of the saxophone family. A♯1 and C♯2 were played by the bass saxophone, E2 to A♯2 by the baritone, C♯3 to A♯3 by the tenor, C♯4 to C♯5 by the alto, and E5 to C♯6 by the soprano sax.

[3] WAV (or WAVE) is a digital audio file format, defined by Microsoft as part of its Resource Interchange File Format (RIFF) framework for the *Windows*™ environment; samples are typically stored as signed 16-bit integers in little-endian byte order.

[4] Publicly available as part of the `cdrtools` package at `http://www.escape.de/users/colossus/cdda2wav.html` (2002-10-01).

[5] Publicly available for the K Desktop Environment at `http://kwave.sourceforge.net` (2002-10-01).

Pitch	Instrument																Count
	bsn	cla	clb	eng	flt	glk	hrn	obo	pno	sax	tpt	trb	vla	vlb	vlc	vln	
A♯1	4.84								3.17	3.09				2.70			
C♯2	5.23		4.29						3.26	2.38				2.71	4.51		
E2	5.84		3.74				1.97		3.47	3.64		1.71		2.75	2.84		43
G2	6.12		3.12				2.47		3.51	3.29		2.07		2.92	5.69		
A♯2	4.23		2.57				2.35		3.37	4.96		1.97		2.71	4.63		
C♯3	4.30		3.01				2.00		3.22	2.99		1.64	2.46	2.39	4.43		
E3	5.19	3.86	2.89	2.87			2.25		3.33	2.77		2.03	2.63	3.16	3.96		
G3	4.76	3.39	3.23	2.87			1.94		3.32	2.54	5.29	2.22	3.64	2.76	4.03	4.08	
A♯3	4.32	3.48	3.52	2.59			2.06	2.89	2.80	2.69	6.32	2.82	2.65	2.90	3.37	3.52	67
C♯4	4.48	3.79	3.38	2.72	3.38		2.05	2.02	2.09	2.33	8.21	2.15	3.03	3.28	3.44	3.78	
E4	4.65	3.75		2.90	3.44		1.77	2.73	3.39	2.78	7.41	3.87	2.78	3.18	3.57	3.74	
G4		3.46		2.73	3.23		1.48	2.54	3.39	3.09	6.51	3.03	3.58		3.56	3.80	
A♯4		3.62		2.79	3.35		1.94	2.91	3.59	2.96	6.37	2.02	4.32		3.10	3.22	46
C♯5		3.67		2.91	4.43		2.25	2.32	3.31	2.60	6.53	2.14	4.28		2.39	3.28	
E5		4.04		3.14	3.74			2.26	2.27	3.03	5.92		3.88		1.93	3.30	
G5		3.55		3.16	3.60	1.00		2.94	3.48	3.11	5.73		3.25		2.00	3.83	
A♯5		3.66			3.85	1.59		2.74	3.79	3.34	5.41		3.49			3.20	34
C♯6		3.66			3.74	1.54		2.57	3.88	3.95	1.85		4.40			3.82	
E6					3.56	1.32		2.31	3.38							3.24	
G6					3.77	1.55			4.26							2.88	8
A♯6					3.35	1.73			5.65							3.01	
Count	11	12	9	10	12	6	12	11	21	18	11	12	13	11	15	14	198

Table 5.1: Durations (in seconds) of the instrument tones selected for testing. The grouping of tones is indicated, with the count of the number of tones in each group.

Group	Harmonics
1	146
2	61
3	31
4	15
5	11

Table 5.2: Number of harmonics retained in the basis spectra for each group of tones.

of harmonics retained in the basis spectra for each group are indicated in Table 5.2.

The groupings were decided somewhat arbitrarily, but subject to a few basic principles:

- The pitch ranges spanned by the lower groups should be larger than those of the upper groups, since the relationship between pitch and frequency is logarithmic, not linear.

- The dividing points between groups should correspond as far as possible with the extremes of the ranges of the selected instruments. For example, the lowest clarinet and English horn tones determine the lower bound of Group 2, and the highest bassoon and string bass tones correspond with the upper bound of the same group.

- A group should include at least two or three tones from a given instrument so that there is a strong likelihood that each of the basis spectra for that group will be used in matching two or more tones. If a group includes a single tone by a given instrument, then the group should at least contain tones of another instrument of the same family. For example, it is acceptable to include the highest English horn pitch, G5, in Group 4 because that tone is likely to be very similar to one or more of the oboe tones included in the same group; similarly, the lone 'cello tone in Group 4 will likely be matched by some of the same basis spectra used to match the viola tones included in the same group. However, it was important that Group 4 begin at G5 rather than A♯5 so that the lower glockenspiel tones were grouped together, since the glockenspiel is distinctly different in timbre from the other selected instruments. Similarly, it was decided to begin Group 5 at G6 rather than at E6 since that group would otherwise have included a solitary oboe tone.

The scheme of selecting tones a minor third apart was used by Horner in his testing of multiple tone matching [41]. Horner tested his multiple wavetable synthesis method (see §2.4.7) on ten English horn tones, twelve trombone tones, fourteen violin tones, and an unspecified number of clarinet, saxophone, viola, and glockenspiel tones; results are given for only the first three instruments. However, Horner does not discuss the problem of avoiding audible arte-

facts due to wrapping around the Nyquist frequency when using harmonic-rich basis spectra selected from lower tones in the synthesis of higher tones; the fourteen violin tones were divided into two sets of seven tones each, but this was done because "matching this extensive space of tones with just six basis spectra did not work" [41, p. 119].

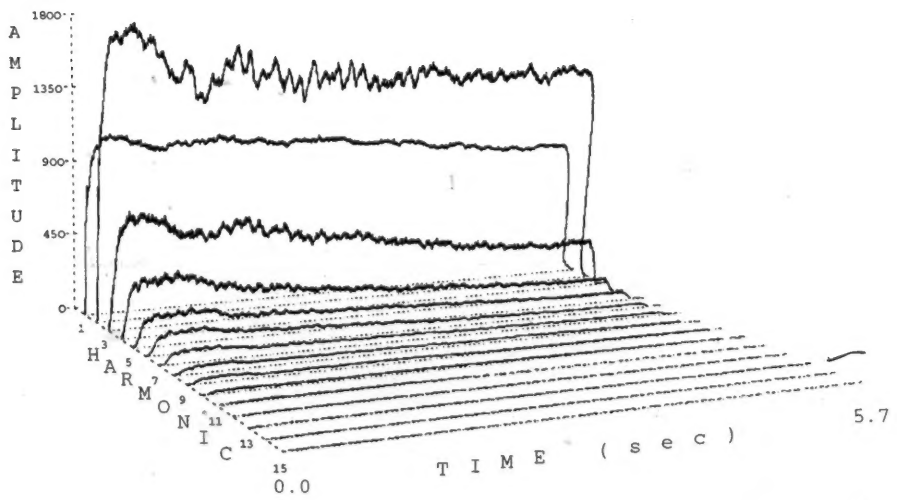
5.2 Results of Waveform Analysis

Most of the tones selected for testing are harmonic tones: their partial frequencies are integer multiples of the fundamental frequency. For example, analysis of the trumpet tone at pitch G5 reveals a clear harmonic structure, as indicated by the even spacing of the dark horizontal frequency lines in the *sonogram* shown in part (b) of Figure 5.1. A sonogram plots frequency as a function of time, with the amplitude of each frequency component represented by the darkness of the line corresponding to that component. The phase vocoder analysis of the same tone, shown in part (a) of the figure, more accurately indicates the amplitude of each partial, but provides a less accurate representation of the frequency of each partial than the sonogram, since the pitch differential calculated by the phase vocoder is ignored in rendering the amplitude × harmonic × time graph.

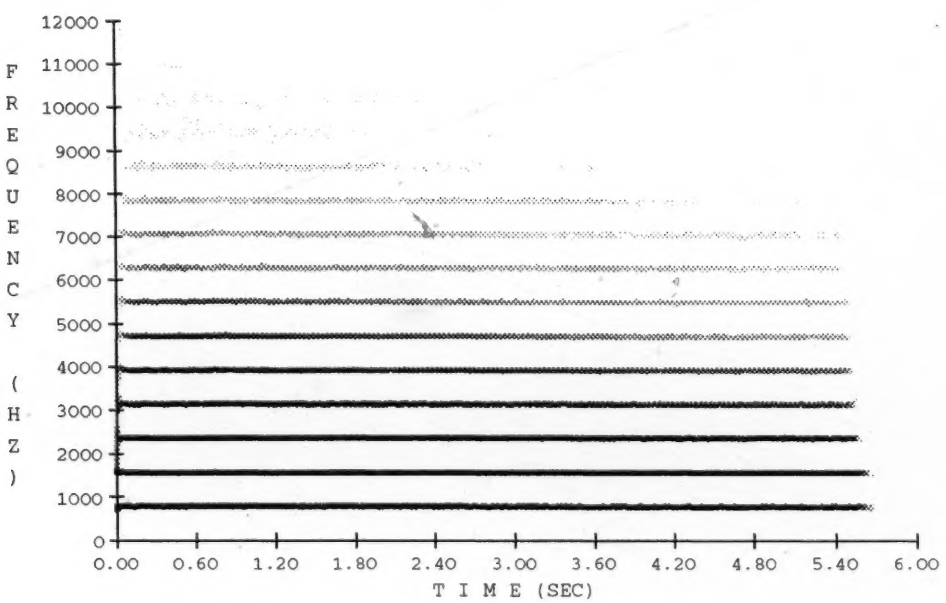
Figure 5.2(a) shows the phase vocoder analysis of a glockenspiel tone at G5, and part (b) shows the sonogram of the first half of the tone. The sonogram clearly indicates that the glockenspiel is an inharmonic tone: the lowest dark line corresponds to the frequency which a listener would identify as the pitch of the tone, 784 Hz; the next higher dark line is at 2419 Hz; the third distinct dark line represents a partial at 4452 Hz; and the fourth dark line, which starts to fade at about 200 ms, is centered at 7626 Hz. Below the fourth line, a lighter line appears to emerge from a broad dark band after about 200 ms, centering at about 6970 Hz. The upper frequencies are 3.1, 5.7, 8.9, and 9.7 times the lowest partial frequency.

The imprecision of the amplitude × harmonic × time representation with respect to frequency is clearly seen by comparing parts (a) and (b) of Figure 5.2. When the frequency differentials of "harmonics" 5 and 6 are taken into account, they actually represent the single partial at 4452 Hz. Because the actual frequency of the partial fell between bins 5 and 6 of the FFT analysis, the amplitude of the partial was divided into two pseudo-harmonics. Similarly, the broad jitter of "harmonic" 9 is an analysis artefact resulting from crosstalk between FFT bins 8, 9, and 10: the energy of the 6970 Hz partial is split between bins 8 and 9 (with most in bin 9), and the energy of the 7626 Hz partial is divided between bins 9 and 10 (mostly in 10).

A clearer analysis results from a technique suggested by Beauchamp [6] for the analysis of percussion sounds: specify a lower frequency of analysis for the phase vocoder than the putative fundamental frequency so that only one modal frequency is aligned with each FFT bin. Figure 5.3 was derived

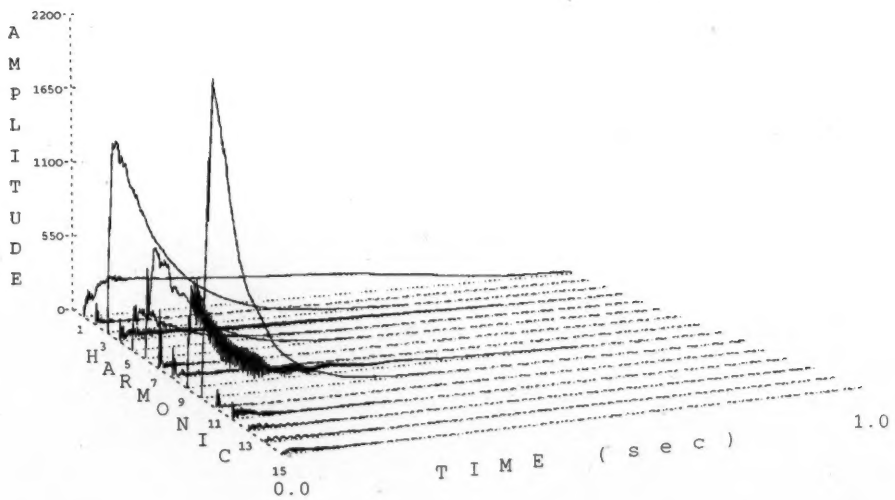


(a) Phase vocoder analysis.

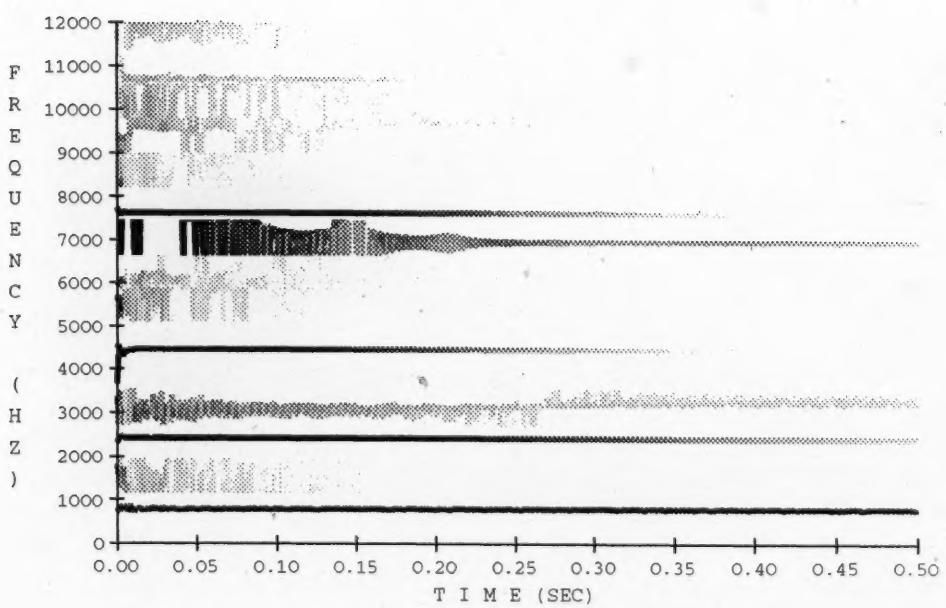


(b) Sonogram.

Figure 5.1: Analysis of a trumpet G5 tone.

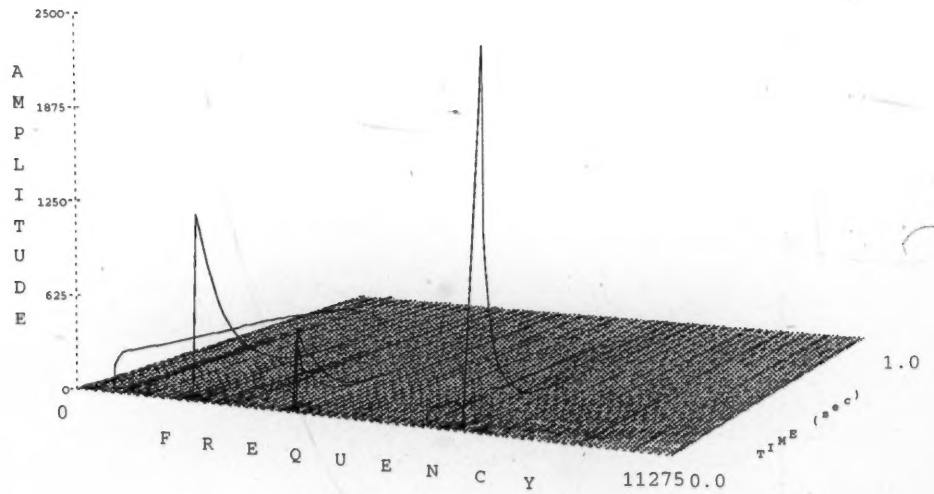


(a) Phase vocoder analysis.

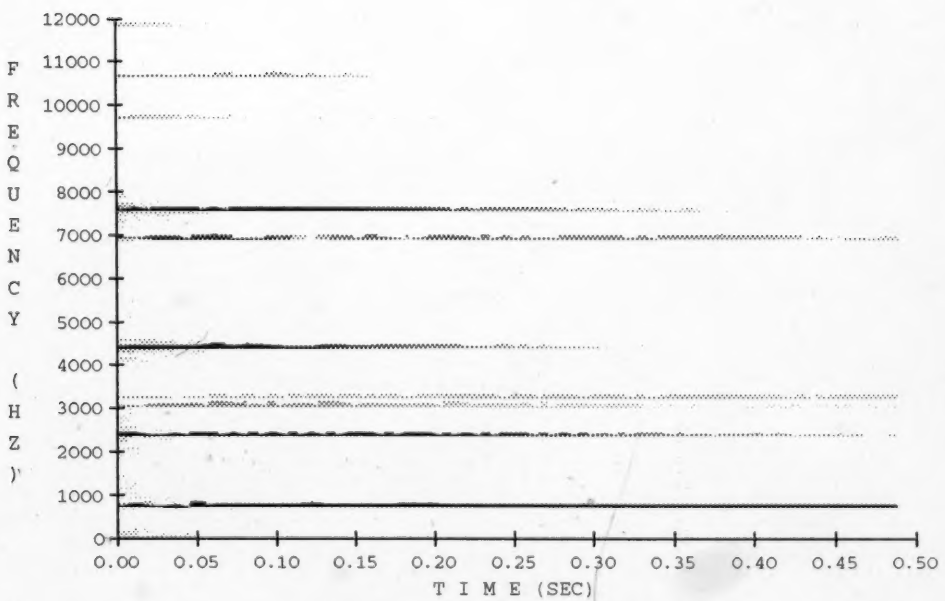


(b) Sonogram of first 500 ms.

Figure 5.2: Analysis of a glockenspiel G5 tone.



(a) Phase vocoder analysis at 41 Hz.



(b) Sonogram of first 500 ms.

Figure 5.3: Analysis of a glockenspiel G5 tone using an analysis frequency of 41 Hz.

by analyzing the glockenspiel G5 tone at 41 Hz, which aligns the 2419 Hz, 6970 Hz, and 7626 Hz partials with FFT bins 59, 170, and 186, respectively, and splits the energy of the partial at 4452 Hz about evenly between bins 108 and 109; the fundamental frequency falls close to bin 19. The associated sonogram also more cleanly (but faintly) delineates the modes of vibration of the glockenspiel's metal bar.

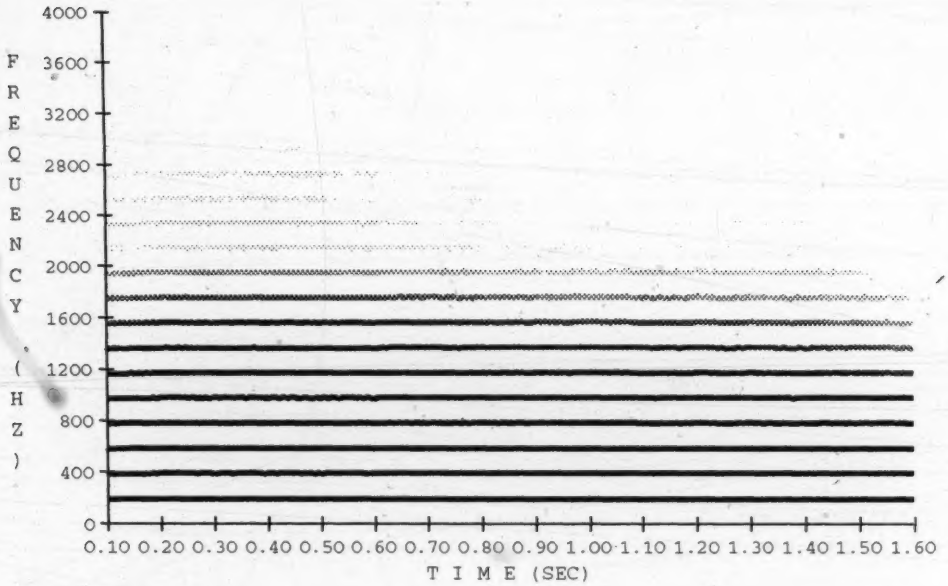
A lesser degree of inharmonicity may be seen in the upper partials of a piano tone, caused by the stiffness of the piano strings [9]. Figure 5.4 compares the sonogram of a French horn G3 tone, which has purely harmonic partials, with that of a piano tone at the same pitch. The inharmonicity may be seen by comparing the positions of the energy lines near 2000, 2400, and 2800 Hz in the horn sonogram with the corresponding lines in the piano sonogram. Although the fundamental frequency of both tones is 196 Hz, the upper partials of the piano tone appear to stretch toward multiples of 200 Hz and then, by the 3200 Hz mark and even more clearly by the 3600 Hz mark, to sharpen even further.

Since synthesis methods based on multiple wavetable additive synthesis are inherently harmonic, allowing only a single frequency differential per spectrum, it is to be expected that optimized multiple wavetable interpolation will produce more realistic approximations of harmonic tones than of inharmonic ones such as those of the glockenspiel and piano. Blackham [9] found that, in listening tests with both musicians and nonmusicians, the jury was able to distinguish between real piano tones and synthetic tones consisting of harmonic partials 86% to 90% of the time; however, both musicians and nonmusicians found real piano tones to be indistinguishable (i.e., misidentified 50% of the time) from synthetic tones using inharmonic partials. As previously discussed in §2.4.9, Horner [42] found that piano tones synthesized by multiple wavetable interpolation were correctly identified in at least 70% of trials of a formal listening test.

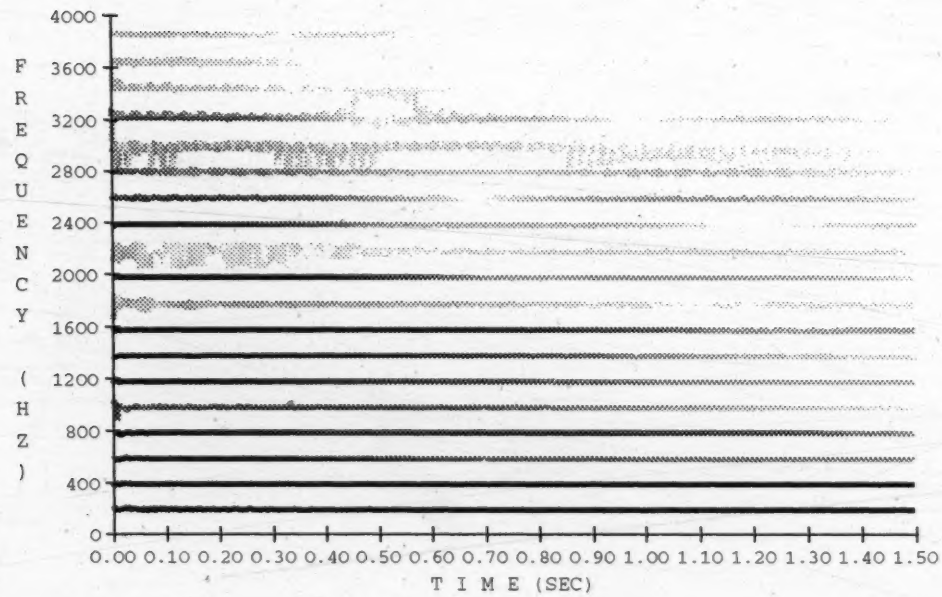
5.3 Breakpoint Selection Results

The phase vocoder analysis of each selected instrumental tone was data-reduced by piecewise-linear approximation using the segment-merging breakpoint selection algorithm. Tones without vibrato (bassoon, clarinet, bass clarinet, glockenspiel, French horn, piano, saxophone, trumpet, and trombone) were approximated with 24 internal breakpoints; the mean Euclidean error (see equation 4.1, p. 66) of each of these approximations is indicated in Table 5.3.

Selecting breakpoints for the tones with vibrato—the English horn, flute, oboe, and strings—was more complicated, and was customized for some tones. A minimum of 48 internal breakpoints was specified for all tones by these instruments, as was a maximum error bound. The segment-merging breakpoint algorithm terminated upon reaching the specified number of breakpoints or upon passing the error bound, whichever occurred first. In general, a lower



(a) Sonogram of a horn G3 tone (first 1.5 sec. after the attack).



(b) Sonogram of a piano G3 tone (first 1.5 sec.).

Figure 5.4: Inharmonicity of the upper partials of a piano tone relative to the harmonicity of a horn tone.

Pitch	Instrument								
	bsn	**cla**	**clb**	**glk**	**hrn**	**pno**	**sax**	**tpt**	**trb**
A♯1	47.0					15.4	24.4		
C♯2	55.6		24.9			15.1	28.0		
E2	58.7		32.3		25.0	13.6	29.5		38.2
G2	45.3		33.1		23.0	8.6	31.5		28.0
A♯2	42.5		29.4		15.8	8.6	20.5		23.4
C♯3	48.8		30.8		24.8	7.7	34.1		20.5
E3	61.8	22.1	27.6		35.6	7.5	21.0		27.8
G3	52.7	21.5	28.5		39.5	6.6	46.8	24.8	39.6
A♯3	79.1	27.1	34.6		23.9	6.2	28.5	20.5	40.9
C♯4	37.0	29.2	28.7		28.3	5.9	27.3	25.0	24.0
E4	44.7	21.4			15.2	4.2	46.2	31.6	19.6
G4		22.8			17.6	5.7	23.1	20.7	22.0
A♯4		22.0			22.1	3.9	28.2	22.0	32.9
C♯5		28.5			29.1	4.9	31.5	32.2	43.3
E5		38.5				5.9	28.8	33.7	
G5		35.9		15.7		2.8	30.7	40.0	
A♯5		31.5		3.0		2.9	41.1	37.7	
C♯6		27.4		4.9		4.3	28.8	38.7	
E6				9.1		3.9			
G6				13.0		3.4			
A♯6				8.8		2.2			

Table 5.3: Mean Euclidean error of piecewise-linear approximations of non-vibrato tones. All tones listed were approximated with 24 internal breakpoints.

error bound was specified for the lower-pitched instruments and a higher error bound for the higher-pitched instruments and tones: 40 for the string bass; 50 for the lower 'cello tones; 60 for most of the upper 'cello tones; 80 for the top 'cello tone, most of the viola tones, and the lower violin tones; and 100 for the flute and most upper violin tones. The exceptions were made for the two highest viola tones, for which an error bound of 50 seemed to suffice, and for two of the upper violin tones, with a bound of 80. An error bound proved to be unnecessary for the English horn and oboe, the tones of both of which are characterized by low-amplitude vibrato; the error level dropped below 40 for all these tones by the time the 48-breakpoint cutoff was reached. The results of the application of the breakpoint-finding algorithms on these tones are presented in Table 5.4, which includes columns indicating the number of internal breakpoints selected and the approximation error for each instrument tone with vibrato.

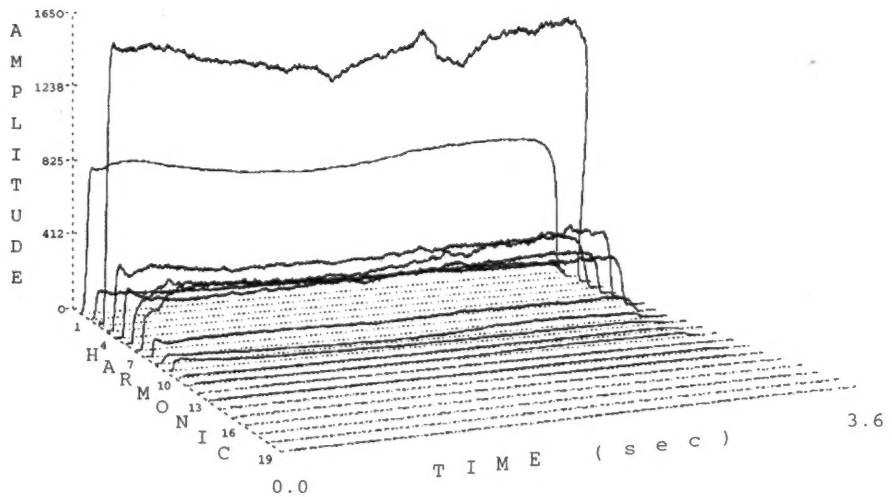
As discussed in §4.2.2, an error measure including an attack bias was used for some flute tones in order that the PLA would model the initial chiff of the tones. Table 5.5 lists the pitches of the flute tones for which an attack bias was used, the duration over which a frame-weighted bias was applied, and the weight of the bias. The error levels reported for these tones in Table 5.4 were calculated according to equation 4.5 and are thus not directly comparable with error levels calculated without an attack bias.

Some experimentation was done on the use of an attack bias when selecting breakpoints in tones with a rapid and strongly peaked attack. For example, Figure 5.5(a) shows a phase vocoder analysis of a clarinet A♯4 tone, which illustrates both a short rise time and the relative weakness of the second harmonic which characterizes the clarinet's low register [1, 32]. Part (b) of the same figure shows that a 20-breakpoint PLA of the clarinet tone does not capture the initial peak of the third harmonic, which occurs slightly later than the initial peaks of the first, second, fourth, fifth, and eighth harmonics. Figure 5.6 illustrates two approaches to solving this problem: part (a) shows a 20-breakpoint PLA using an attack bias of weight 2 for frames in the initial 140 ms of the tone; part (b) shows a 24-breakpoint PLA without an attack bias. On the basis of this experimentation, it was decided that the use of additional breakpoints was preferable to the use of an attack bias to model high-amplitude features of the attack segment, due to the sensitivity of the latter method to the parameters of the bias (weight and duration). It was also concluded that 24 is a reasonable minimum number of internal breakpoints to use for piecewise-linear approximation of non-vibrato tones.

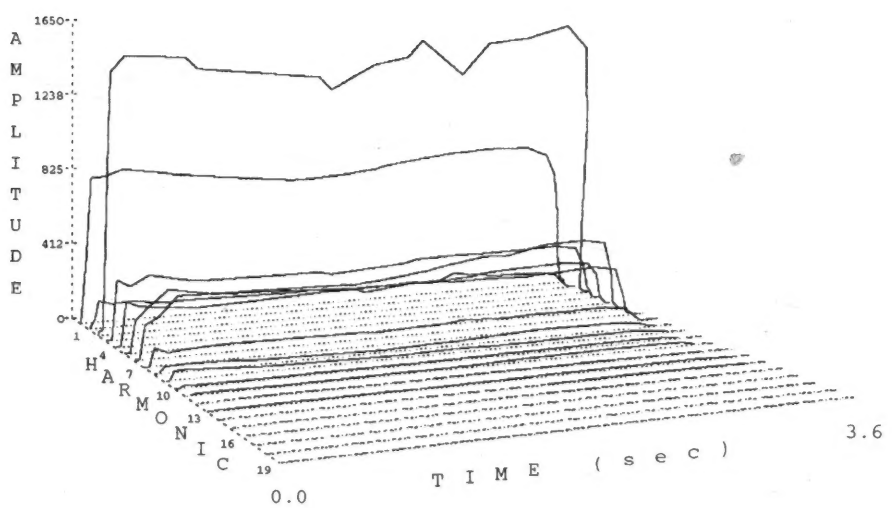
Some of the results of the segment-merging breakpoint selection algorithm were shown previously in Figures 3.4, 3.5, 3.12, and 4.1–4.6, which illustrate the piecewise-linear approximation of tones with a sustain section and tones with vibrato. Figure 5.7 illustrates the approximation of an amplitude envelope with a short attack section and a long decay, exemplified by the same piano tone for which a sonogram was given in Figure 5.4; Figure 5.8 shows the same analyses in two-dimensional views.

Pitch	Instrument													
	eng		flt		obo		vla		vlb		vlc		vln	
	Bkpts	Error	Bkpts	Error	Bkpts	Error	Bkpts	Error	Bkpts	Error	Bkpts	Error	Bkpts	Error
A♯1									48	33.0				
C♯2									55	40.7	51	50.8		
E2									48	28.3	48	37.0		
G2									48	32.5	48	35.0		
A♯2									52	40.8	48	46.1		
C♯3							55	80.1	48	32.5	54	50.8		
E3	48	13.0					108	80.5	48	40.1	68	50.1		
G3	48	22.7					140	80.1	53	40.1	48	33.7	48	57.0
A♯3	48	22.7			48	24.3	113	80.1	48	29.8	48	39.2	99	80.1
C♯4	48	17.5	48	76.2	48	13.0	97	80.2	57	40.3	57	50.3	123	80.1
E4	48	21.2	64	100.4	48	29.2	130	80.5	69	40.4	65	50.0	68	80.0
G4	48	34.5	63	100.6	48	27.7	98	80.5			81	61.0	88	80.2
A♯4	48	18.6	92	101.2	48	36.5	117	80.3			78	60.5	66	80.6
C♯5	48	9.8	85	100.0	48	31.7	98	80.4			91	60.4	122	80.8
E5	48	24.2	65	100.0	48	38.7	174	80.2			113	60.0	112	80.8
G5	48	29.5	71	101.0	48	31.8	173	80.0			121	80.5	104	100.7
A♯5			73	101.6	48	39.0	105	50.1					122	100.1
C♯6			62	101.5	48	35.1	87	50.3					179	100.1
E6			68	100.0	48	35.6							108	80.2
G6			53	100.9									87	80.4
A♯6			85	101.2									95	100.7

Table 5.4: Mean Euclidean error of piecewise-linear approximations of tones with vibrato. The number of internal breakpoints selected and the approximation error are indicated for each tone.

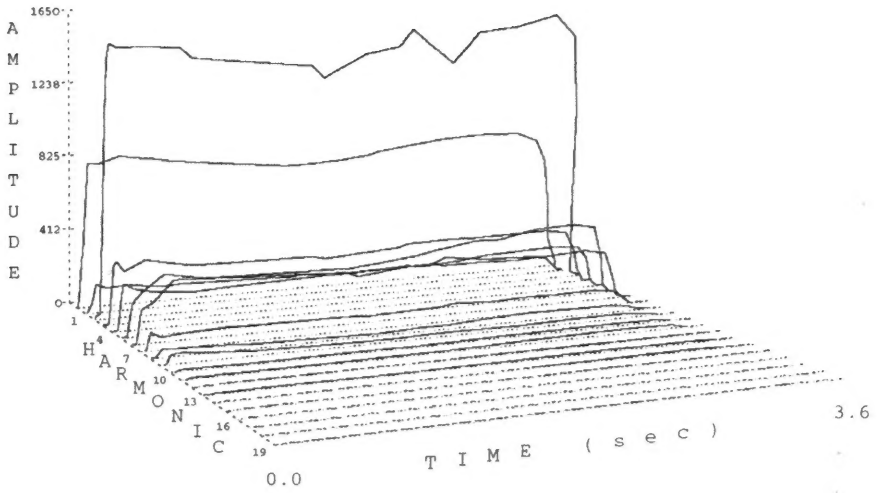


(a) Phase vocoder analysis.

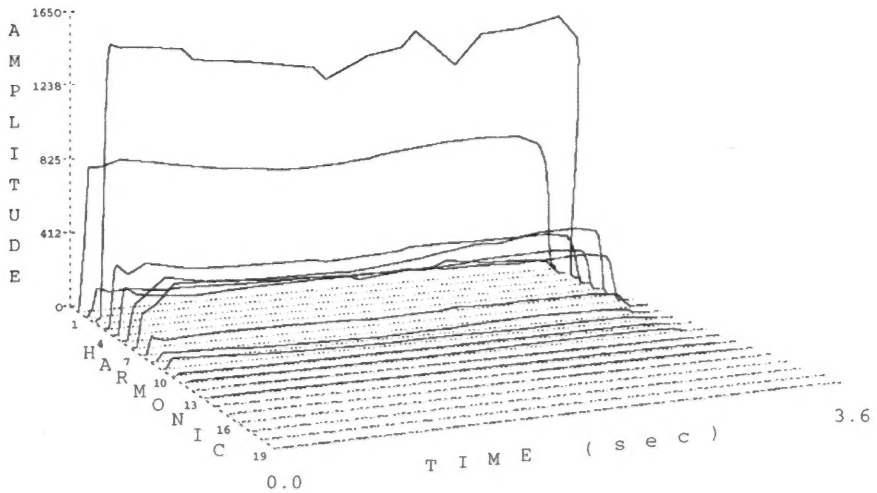


(b) Piecewise-linear approximation with 20 breakpoints.

Figure 5.5: Analysis and 20-breakpoint PLA of a clarinet A♯4 tone.

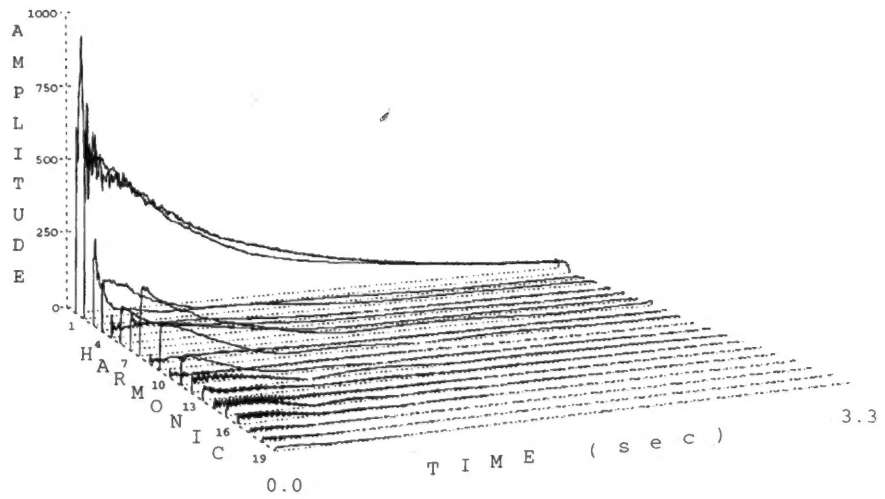


(a) Twenty-breakpoint PLA with attack bias.

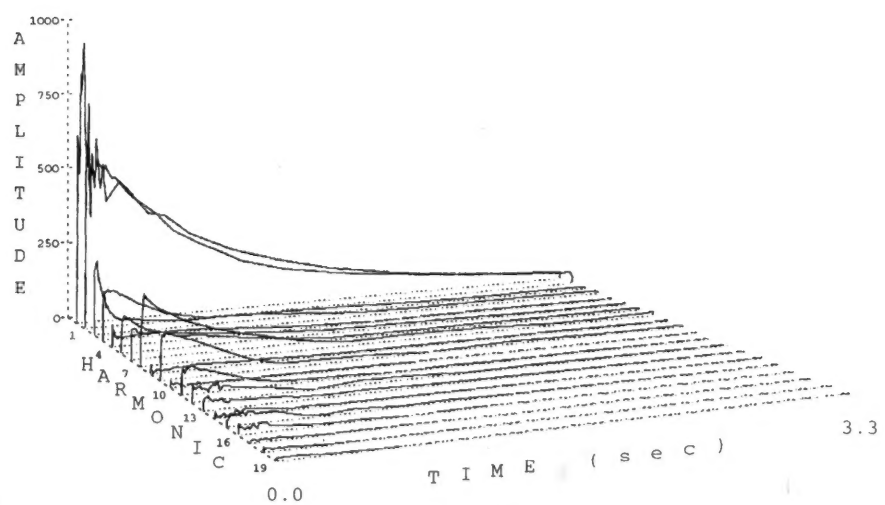


(b) Twenty-four-breakpoint PLA without attack bias.

Figure 5.6: Comparison of two methods to model the initial peak of a clarinet A♯4 tone.

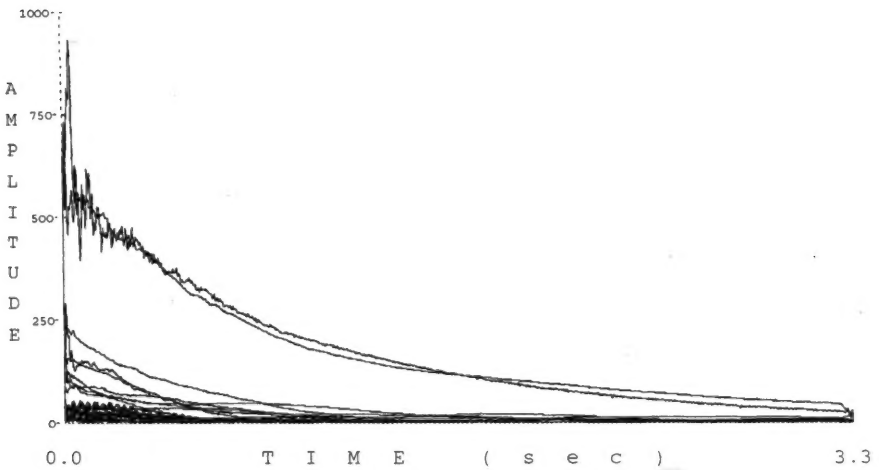


(a) Phase vocoder analysis (first 20 harmonics).

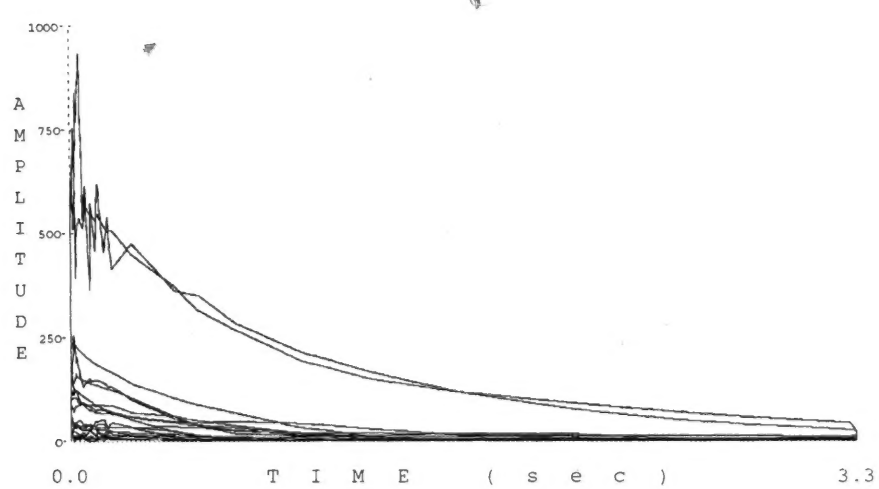


(b) Piecewise-linear approximation with 24 internal breakpoints.

Figure 5.7: Phase vocoder analysis and 24-breakpoint PLA of a piano G3 tone.



(a) Phase vocoder analysis (first 20 harmonics).



(b) Piecewise-linear approximation with 24 internal breakpoints.

Figure 5.8: Phase vocoder analysis and 24-breakpoint PLA of a piano G3 tone (two-dimensional view).

Pitch	Duration	Weight
C♯4	85 ms	4
E4	165 ms	4
A♯4	50 ms	12
C♯5	50 ms	8
E5	65 ms	8

Table 5.5: Duration and weight of a frame-weighted attack bias applied to selected flute tones.

From the total of 602,957 analysis frames for which the phase vocoder extracted a harmonic spectrum, 9034 (or 1.5%) were selected as breakpoints, an average of 45.6 breakpoints per tone.

5.4 Wavetable Bank Selection Results

5.4.1 Clustering Results

The clustering program `AutoClass C` was run on each group of normalized breakpoints using the single-normal covariant normal model. All the harmonics which were retained in the spectra of each group (as indicated in Table 5.2) were used as attributes.[6] The best two clusterings found in 250 tries were recorded for each group. The results of the clustering of the breakpoint spectra[7] from each group of tones are summarized in Table 5.6. Overall, considering both of the best two clusterings for each group, the clusterer identified an average of 1.3 classes per instrumental tone, with an average of 35.9 breakpoints per class.

To illustrate the results of clustering with respect to the spectral envelopes of individual tones, Table 5.7 shows, for four example tones, to which class each breakpoint spectrum was assigned in the best clustering of the breakpoint spectra in Group 2. Each breakpoint is identified by its time index in seconds relative to the start of the tone at time 0.

The horn and sax tones, which were previously seen in Figures 3.2–3.5, illustrate a commonly occurring pattern in which all or most of the spectra from the sustain portion of a tone are clustered together, while spectra from the attack and release segments are assigned to various other clusters. The

[6] `AutoClass C` was revised to handle more than its default maximum of 20 attributes.

[7] At an earlier stage of breakpoint selection than that reported in §5.3, the segment-merging algorithm was applied to the trumpet tones with a minimum of 48 breakpoints and an error bound of 20 as parameters, resulting in the selection of an average of 73.5 breakpoints per trumpet tone. `AutoClass C` was run on groups of breakpoint spectra including this larger proportion of trumpet spectra. Subsequent testing revealed that it was not necessary to model the shimmer of the trumpet tone to this level of detail, and the trumpet tones were re-approximated with only 24 breakpoints per tone to be consistent with the other tones without vibrato. The breakpoint matching algorithm was applied to the smaller trumpet PLA's.

Group	Tones	Break-points	Classes	Breakpoints per Class
1	43	1347	44	30.6
			46	29.3
2	67	3059	74	41.3
			85	36.0
3	46	2632	65	40.5
			66	39.9
4	34	2125	82	25.9
			48	44.3
5	8	416	12	34.7
			11	37.8

Table 5.6: Number of classes of normalized breakpoint spectra and average number of breakpoints per class in each group of tones as identified by clustering. Results are shown for each of the best two clusterings found.

piano tone, also featured in Figures 5.4(b) and 5.7–5.8, shows spectra passing through a sequence of classes as the tone decays from an extremely quick attack.

The viola tone was approximated with 140 breakpoints, as illustrated in Figure 5.9. The sustain portion of the tone (which includes vibrato) begins with about a half-second section of spectra assigned to class 10, followed by more than a second of mixed class 10 and class 12 spectra, concluding with about a second of class 12 spectra. Of particular interest from the point of view of the premises of optimized multiple wavetable interpolation are the spectra of the attack and release segments: the attack segment includes two spectra which were clustered along with two spectra from the release of the saxophone tone in class 27, and the release segment includes spectra from classes 1, 5, 15, and 18, which also include spectra from the decay of the piano tone, the start of the saxophone attack, the release of the horn tone, and the final breakpoint of the sax tone, respectively.

Table 5.8 presents the clustering of the breakpoint spectra in Group 1 from the opposite perspective, showing the distribution of spectra from different classes across various instruments and tones. For each tone in Group 1, the classes to which spectra from that tone were assigned are listed.

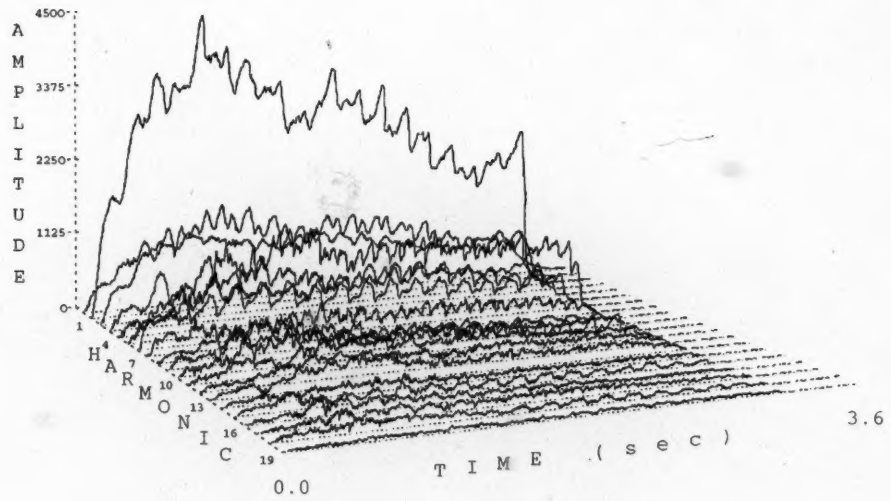
Close study of the clustering results shows that the algorithm has identified various types of classes of spectra:

- Some classes include spectra from many different instruments. For example, classes 2, 4, 6, and 18 each contain spectra from all nine instruments in Group 1; class 21 contains spectra from the attack portions of six instruments; and classes 10, 11, 13, and 25 appear in five different instrument lists.

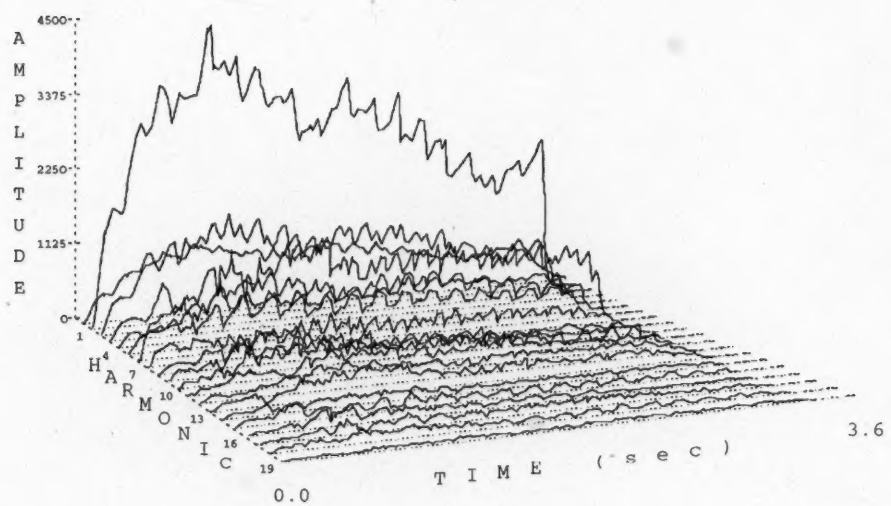
- Some classes are restricted to spectra from a single instrument (or almost

Hrn G3		Pno G3		Sax G3		Vla G3	
Time	**Class**	**Time**	**Class**	**Time**	**Class**	**Time**	**Class**
0.015	0	0.003	72	0.020	5	0.021	7
0.028	9	0.013	72	0.041	57	0.039	27
0.043	0	0.018	72	0.054	57	0.073	10
0.059	20	0.033	17	0.120	57	⋮	⋮
0.151	20	0.049	17	0.261	57	0.330	8
0.200	63	0.056	17	0.562	57	0.364	27
0.276	63	0.074	17	0.680	57	0.390	10
0.384	20	0.079	17	0.828	57	⋮	⋮
0.588	20	0.095	17	0.907	57	0.853	10 & 12
0.634	20	0.102	17	1.025	57	⋮	⋮
0.783	20	0.130	17	1.130	57	2.100	12
0.813	20	0.143	17	1.176	57	⋮	⋮
0.952	20	0.161	17	1.331	57	3.020	12
1.003	20	0.245	17	1.380	57	3.046	58
1.125	20	0.427	4	1.526	57	3.070	8
1.212	20	0.527	4	1.738	57	3.093	8
1.264	20	0.685	4	1.771	57	3.114	8
1.315	20	0.959	4	1.945	57	3.137	1
1.376	20	1.237	4	1.975	57	3.171	1
1.489	20	1.618	1	2.006	57	3.197	1
1.591	9	2.109	1	2.037	27	3.247	15
1.701	9	2.521	26	2.060	4	3.275	5
1.791	15	2.838	26	2.078	27	3.405	5
1.824	15	3.259	26	2.190	18	3.488	18

Table 5.7: The classes to which the breakpoint spectra of four example tones are assigned by clustering. Each spectrum is identified by its time index in seconds.



(a) Phase vocoder analysis.



(b) PLA using 140 breakpoints.

Figure 5.9: Analysis and piecewise-linear approximation of a viola G3 tone.

Pitch	Instrument								
	bsn	**clb**	**hrn**	**pno**	**sax**	**trb**	**vla**	**vlb**	**vlc**
A♯1	2, 13, 18, 21, 26			30, 42	2, 10, 34, 42			2, 4, 6, 11, 13, 16, 18, 21	
C♯2	4, 18, 25, 40	6, 10, 13, 18, 19, 43		2, 10, 24, 42	2, 4, 6, 21, 25, 29			2, 3, 4, 6, 7, 8, 11, 13, 18	4, 6, 13, 18, 20, 21, 33
E2	2, 3, 4, 18, 39	2, 6, 8, 13, 19, 43	2, 3, 4, 5, 6, 10, 11, 14, 22, 25	2, 6, 10, 12, 21, 24, 42	2, 6, 13, 25, 28	2, 6, 10, 22, 36		2, 6, 8, 13, 18	2, 4, 6, 11, 13, 15, 18
G2	2, 4, 11, 18, 27	2, 4, 6, 10, 13, 21, 41	4, 6, 9, 11, 14	2, 6, 12, 21, 42	2, 3, 13, 25, 32, 40	2, 3, 4, 10, 11, 22		2, 6, 11, 13, 18, 23, 38, 43	2, 4, 6, 11, 17, 18
A♯2	3, 4, 6, 18	2, 6, 8, 10, 13, 18, 40	2, 4, 6, 9, 11, 18	2, 4, 6, 10, 12, 21, 43	0, 2, 6, 10, 31	2, 4, 5, 6, 9, 11, 14, 25		1, 3, 4, 6, 11, 18, 21	0, 2, 4, 11, 18
C♯3	3, 4, 6, 18	2, 6, 10, 18, 35	6, 9, 11, 18	2, 4, 6, 12, 18, 21	2, 4, 8, 10, 13, 18, 37	3, 4, 9, 11, 14, 18	2, 3, 4, 5, 6, 18, 21, 25, 39	1, 2, 4, 6, 11, 13, 18	0, 4, 6, 11, 18

Table 5.8: The classes to which spectra from each tone in Group 1 were assigned by clustering.

to a single instrument) but from multiple tones of that instrument. For example, class 0 consists of 81 spectra from two 'cello tones plus a single saxophone spectrum; classes 1, 7, and 8 are restricted to spectra from one or two bass tones except for a few bass clarinet and sax spectra in class 8; classes 19 and 24 consist of spectra from only two bass clarinet and piano tones, respectively; class 3 consists mostly of bassoon spectra; and classes 9 and 12 are confined to horn and piano spectra, respectively, except for two trombone spectra in class 9.

- Some classes are specifically focused, but are not restricted to a single instrument. For example, class 5 is a collection of spectra from the horn E2, the trombone A♯2, and the viola C♯3; classes 14 and 22 each consist of horn and trombone spectra; the spectra in class 39 are from the bassoon E2 tone and the viola C♯3 tone; and class 40 consists of spectra from the bassoon C♯2 plus two other spectra.

- Some classes consist entirely of spectra from a single instrument at a single pitch. For example, class 15 contains only spectra from the 'cello E2 tone; class 16 consists of string bass spectra from the A♯1 tone; class 17 is restricted to 'cello G2 spectra; and classes 20, 23, 26–38, and 41 are similarly focused.

These results confirm that, as speculated in §1.2.1, some spectra of a given instrumental tone are more similar to spectra from other tones of different instruments than they are to other spectra from the same instrument and tone.

5.4.2 Hand-Tuning of Wavetable Banks

As discussed in §4.3.3, the class representatives for the classes in each group were used as a wavetable bank for an initial test of breakpoint matching to determine if the wavetable bank could be improved by adding spectra to reduce any unusually high matching errors or if it could be reduced in size by removing little-used basis spectra.

When the detailed (breakpoint-by-breakpoint) matching results were studied for those tones which had unusually high overall error levels, it was found that the breakpoints with high matching error were typically consecutive and their spectra belonged to the same class. In these cases, one of the high-error breakpoint spectra (typically, the spectrum nearest to the centroid of the high-error spectra) was selected from each problem tone and added to the wavetable bank.

For example, after an initial 4-oscillator "3+1" matching of the tones in Group 1 using the 44 class representatives from the best clustering as a wavetable bank, the six highest-error matchings were the bassoon E2 tone, with a root-mean-square matching error of 779.8; the bassoon A♯2, 463.7; the bass clarinet A♯2, 293.3; the bassoon C♯3, 289.0; the viola C♯3, 240.8; and the

trombone A♯2, 207.2. The average error for all tones in Group 1 using this wavetable bank was 134.8, with a standard deviation of 126.3.

When an additional basis spectrum was added to the wavetable bank for Group 1 from each of these tones except the viola, the error levels dropped to 101.4, 76.2, 119.6, 47.3, and 51.9, respectively; these improvements, along with minor changes (both upward and downward) in the error levels of other tones, reduced the average RMS error to 94.6, with a standard deviation of 37.7. No additional spectrum was added from the viola C♯3 because it was not obvious which spectrum should be added to the wavetable bank to reduce the tone's matching error: no spectrum appeared to be far distant (in a Euclidean sense) from the class representatives of the relevant classes.

Statistics were also gathered on the number of times each basis spectrum in the wavetable bank was used in matching the breakpoint spectra of the tones in each group. For example, each wavetable of the bank for Group 1 consisting of the 44 class representatives was used an average of 109.0 times in a 4-oscillator "3+1" breakpoint matching; usage of individual basis spectra ranged from a low of 6 to a high of 430 uses. Basis spectra usage in a 4-oscillator "2+1" search yielded similar results, with the same least-used wavetable being used only 4 times. The second-least-used wavetable was used 21 times in the "3+1" search and 16 times in the "2+1" search.

On the basis of these usage statistics, it was decided to remove the least-used wavetable from the wavetable bank, resulting in a final wavetable bank for Group 1 of 48 wavetables.

Usage statistics were also used to verify that the added basis spectra were well chosen. It was found that the five added wavetables were used an average of 118.8 times each in a 4-oscillator "3+1" matching, ranging from a minimum of 39 to a maximum of 246 uses; this compares with an overall average usage of 98.3 per wavetable. A 4-oscillator "2+1" matching showed even greater usage of the added wavetables: an average of 124.4 uses, compared to an overall average of 97.6.

Different methods of hand-tuning of wavetable banks appeared to work best for the different groups. For Group 2, a single hand-picked wavetable was added and the single least-used wavetable was removed from the bank of class representatives, leaving the bank at 74 wavetables. This change yielded a 48% improvement in the RMS matching error of the bass clarinet A♯3 tone, from 300.3 to 155.1. As in Group 1, the viola tones remained problematic, with error levels ranging from 182.4 to 317.1, compared to an average matching error for Group 2 of 87.7. Testing of a wavetable bank with an additional viola spectrum showed almost no improvement in matching error levels for viola tones.

The wavetable bank for Group 3 consisting only of the class representatives gave good results, although the viola tones again had higher matching errors than the other tones of the group. However, one wavetable was used only five times in a 4-oscillator "3+1" search and 22 times in a "2+1" search, and was removed from the wavetable bank, leaving a bank of size 64.

Group	Tones	Wavetables in Bank	Wavetables per Tone	Bank Size (bytes)
1	43	48	1.1	28032
2	67	74	1.1	18056
3	46	64	1.4	7936
4	34	48	1.4	2880
5	8	12	1.5	528
Total	**198**	**246**	**1.2**	**57432**

Table 5.9: Number of basis spectra in each wavetable bank and average number of wavetables per tone for each group.

The ratio of the number of clusters in the best clustering found by `AutoClass C` to the number of tones in the group was significantly higher for Group 4 than for Groups 1 to 3: 2.4, compared to 1.0, 1.1, and 1.4, respectively. Usage statistics for a 4-oscillator "3+1" matching showed an average usage of only 69.2 per wavetable, and five of the wavetables were used only 14 to 17 times each. For these reasons, it was decided to use the class representatives of the second-best clustering as the wavetable bank for Group 4, the 48 wavetables of this alternative bank representing a more reasonable ratio of wavetables to tones of 1.4. As a result, the average error of a 4-oscillator "3+1" matching increased 37% from 35.8 to 49.2. Much of this increase was due to large increases (2.5 and 2.6 times) in the already large error of two glockenspiel tones. When these two tones are excluded from the average error calculation, the increase is a more modest 12%. The difference in the tones resynthesized using the two wavetable banks was not audible, perhaps due to the obviously artificial quality of both synthesized tones.

The class representatives of the best clustering of the tones in Group 5 were used without additions or deletions, since the 1.5 ratio of wavetables to tones seemed reasonable for such a small group of tones and no wavetables were significantly under-utilized.

A summary of the final size of each wavetable bank and the ratio of the size of each bank to the number of tones in its corresponding group is provided in Table 5.9. The size of each bank in bytes is also indicated, assuming that harmonic amplitudes are represented as 4-byte floating-point values.

The overall ratio of wavetables to tones of 1.2 is approximately twice that of Horner's experiment [41] with multiple-tone matching of 10 English horn, 12 trombone, and 14 violin tones with five, six, and two sets of five wavetables, respectively (see §2.4.7). However, Horner's multiple wavetable matching requires the use of all five or six wavetables simultaneously (i.e., five or six oscillators), while the method proposed here allows the use of as few as two oscillators and typically three or four oscillators, each with flexible access to a larger wavetable bank. As mentioned in §5.1, Horner does not address the necessity of reducing the number of harmonics in the wavetables used for synthesis in order to avoid artefacts due to upper harmonics wrapping around the

Nyquist frequency. If the English horn and trombone tones had been divided into at least two groups each and the violin tones into at least three groups (compared to the four used in this research), then the ratio of wavetables to tones would have been approximately equal to the ratio reported here.

5.5 Breakpoint Matching Results

The proposed breakpoint matching method begins with a search for an n-table match to the spectrum at each breakpoint, $n \leq N_{\text{osc}}$, where N_{osc} is the number of available oscillators. Two search strategies were tested: exhaustive search and a genetic algorithm. In the general case, the search is a multi-level search consisting of an initial exhaustive or GA search followed by a second-level exhaustive search to augment the size of the initial match. The results (time and matching error) for single- and multi-level exhaustive search are presented in §5.5.1; those for searches using a GA are presented in §5.5.2.

The multi-level search for a match to each breakpoint spectrum is followed by an oscillator assignment optimization stage which uses a shortest-path algorithm on a vertex-weighted directed acyclic graph. The results of optimization of matches found by multi-level exhaustive search are presented in §5.5.3 and of those found with an initial GA search in §5.5.4. The results of the optimization stage are presented separately from the results for the search stage because a search for an initial match need only be conducted once; the match found by a given search (such as a "3+1" exhaustive search or a "2+1" GA search) can be used in multiple optimizations for different numbers of oscillators.

In §5.5.5, the results of using Horner's multiple wavetable interpolation matching method are presented for comparison with those of the proposed new method.

All timing information was derived by executing `C++` programs on a 500 MHz Intel Pentium II Celeron-based system with 256 MB of memory and 750 MB of swap space, running the Linux operating system, kernel version 2.4.5. Programs were compiled with the GNU `g++` compiler, version 2.95.3, against the `Glibc` C library, version 2.2.5 (Linux `libc` 6), and the `libstdc++-3` Standard C++ library, version 2-2-2.10.0.

5.5.1 Multi-Level Exhaustive Search Results

First-Level Exhaustive Search

The time required to perform an exhaustive search of depth m (i.e., a search for the best possible m-wavetable match for a given breakpoint spectrum) is predominantly due to the time required to perform an LUP decomposition of each matrix of basis spectra harmonic amplitudes and to find a least-squares solution of the system shown in equation 3.2 for each breakpoint spectrum. The number of LUP decompositions performed in the course of an exhaustive

search for an m-table match where the tables are drawn from a wavetable bank of size N_{WT} is

$$\binom{N_{\mathrm{WT}}}{m} = \frac{N_{\mathrm{WT}}!}{m!(N_{\mathrm{WT}} - m)!} \tag{5.1}$$

and the number of least-squares solutions to be found is

$$\binom{N_{\mathrm{WT}}}{m} N_{\mathrm{bkpt}} \tag{5.2}$$

where N_{bkpt} is the number of breakpoints in the current tone. A single LUP decomposition takes about the same amount of time as a single least-squares solution, and both are dependent on the number of harmonics in the basis spectra of the wavetable bank.

Table A.1 on page 153 in Appendix A indicates the amount of time (in seconds) required to perform a single-level exhaustive search of depths 1 through 4 for matches drawn from the wavetable banks for Group 1 and the RMS matching error of each best match relative to the breakpoint spectrum being matched. The search times and matching errors for Groups 2–5 are available in the accompanying technical report [62].

Augmented Search

The time required for a second-level exhaustive search of a given depth is greater than that for a first-level exhaustive search of the same depth because the augmenting search performs an LUP decomposition and N_{bkpt} least-squares solutions for each unique set of wavetables which have been found to form best matches to the breakpoint spectra of the current tone by the initial search. If the number of unique sets of matching wavetables is N_{uniq}, then the number of LUP decompositions performed at this stage of the search is

$$\binom{N_{\mathrm{WT}} - m}{n - m} N_{\mathrm{uniq}} \tag{5.3}$$

and the number of least-squares solutions at this stage is

$$\binom{N_{\mathrm{WT}} - m}{n - m} N_{\mathrm{uniq}} \cdot N_{\mathrm{bkpt}} \tag{5.4}$$

where, as above, m is the depth of the initial search, n is the desired size of the match after augmentation, N_{WT} is the number of wavetables in the relevant wavetable bank, and N_{bkpt} is the number of breakpoints in the current tone.

Typically, a search to augment a larger first-level match will take longer than one to augment a smaller initial match, since a larger match is more specific; for example, the same pair of wavetables may be the best match to all the breakpoints in the sustain portion of some tone, but a given 3-table set might only match a few different breakpoint spectra.

Tables A.2 to A.3 (pp. 154–155) indicate the time (in seconds) to perform a two-level search for each tone and the RMS matching error of the resulting matches to the breakpoint spectra of each tone in Group 1. The first table presents the results for searches which augment an initial search with a single additional wavetable, and the second, the results of augmenting with two additional wavetables. The augmented search results for the other groups are available in the technical report [62].

The first column of data for each category of augmented search, headed "T_{aug}," indicates the time required to augment an existing match found by a first-level exhaustive search, the results of which can be cached and reused for subsequent augmenting searches. Figures in the second column of each group, "T_{srch}" (search time), include both the time required by the initial exhaustive search and that for the augmenting search. These aggregate times are shown to facilitate comparison with the times and error levels of Table A.1. For example, it may be seen that the time required for a "2+1" augmented search will typically be less than that for a single depth-3 search, but may result in matches of size 3 which have almost the same matching error as those found by exhaustive search. Furthermore, the optimizer may be able to find a lower-error solution when beginning with the matches found by a "2+1" search than with the better matches found by exhaustive search, due to use of fewer different wavetables in the "2+1" matches (which reduces the need for fade-ins and fade-outs and maximizes oscillator use).

Results are not shown for "1+1" searches, but they predictable: search times are negligible, and matching errors are about 30% to 80% greater than those for "2+1" matches.

It is also possible to conduct an augmenting search in stages, which further reduces the required search time by pruning the search tree. For example, a "2+3" search for 5-wavetable matches would likely require an unreasonable amount of computation, since an exhaustive search of depth 3 would have to be conducted for each of the 2-wavetable sets found by the initial search; however, a "2+2+1" search requires only about 10% more time than a "2+2" augmented search but produces matches which are about 20% better.

Summary

A summary of the results of multi-level exhaustive search is presented in Table 5.10, which lists the mean execution time (in seconds) and the mean RMS error for each type of search and each group of tones, in increasing order by time. The table confirms that, in general, a search which takes more time (i.e., traverses more of the search tree) produces better results, but there are some cases which do not conform to this general model. For example, for a 5-table match, a "3+2" search is to be preferred to a "4+1" search, since it produces approximately the same or better results in significantly less time.

The same information is illustrated in graph form in Figure 5.10, which makes clear the stepped nature of the results: after the lone data point for a

Type of Search	Group									
	1		2		3		4		5	
	Time	Error	Time	Error	Time	Error	Time	Error	Time	Error
1+0	0.15	196.4	0.14	251.7	0.08	191.8	0.04	184.5	0.01	206.0
1+1	1.32	134.4	1.73	142.0	1.54	84.0	0.62	80.8	0.07	128.2
2+0	6.41	132.9	9.17	138.4	4.51	82.2	1.42	79.0	0.06	127.7
1+2	7.20	101.6	10.8	94.4	9.34	46.4	4.11	51.0	0.35	84.3
2+1	12.3	100.9	18.4	92.4	12.3	45.8	5.08	50.4	0.34	84.1
3+0	145	98.6	317	86.0	134	41.8	32.2	45.2	0.29	83.9
3+1	157	82.1	339	62.8	151	28.6	40.9	33.4	0.91	55.1
2+2	181	81.4	441	60.7	328	25.9	111	29.3	1.74	53.0
2+2+1	199	70.5	472	47.3	352	20.0	123	23.5	2.81	30.3
3+2	477	68.6	1264	44.4	783	18.5	269	21.5	3.55	30.2
4+0	2114	79.4	7450	56.9	2706	24.2	470	27.3	0.87	48.6
4+1	2132	69.2	7482	45.1	2730	19.1	482	21.2	1.81	30.3

Table 5.10: Summary of multi-level exhaustive search results.

1-table match, the curve for each group steps down to an almost horizontal plateau representing the 2-table matches, followed by another for the 3-table matches. Two more horizontal plateaus can be seen if the penultimate data point (the result of the "4+0" search) is viewed as connected to the other 4-table matches ("3+1" and "2+2") instead of being linked vertically to the final point, which can be seen as the rightward extension of the step begun by the data points for "2+2+1" and "3+2." The graph also makes it clear that the single-level exhaustive searches ("2+0," "3+0," and "4+0") give very little improvement over the two-level (augmented) searches of the same size but cost significantly more. (Note that the time axis is logarithmic in the figure.)

It should be noted that lower matching errors at this stage do not necessarily yield lower matching errors after oscillator assignment optimization. In some cases, the initial matches are further augmented after the overlapping stage. However, the primary reason that there is no direct relationship between the quality of initial matches and the quality of oscillator assignments is that matches which are highly specific to their respective breakpoint spectra are less likely to be used in a final set of matches than more general ones, due to the need to fade in and out any wavetables which change from one breakpoint to the next. The matching results presented in this section should be compared with the optimization results presented below in §5.5.3.

5.5.2 Genetic Algorithm Search Results

The use of a genetic algorithm instead of an initial exhaustive search was tested only on Group 1, the results of which are shown in Tables A.4–A.7 (pp. 156–159). The first two tables summarize the results of a thorough genetic algorithm search, with a population size of 100 and termination on convergence after 50 generations; the latter two tables are for a quick version of the GA search, with population size 50 and 25 generations to convergence.

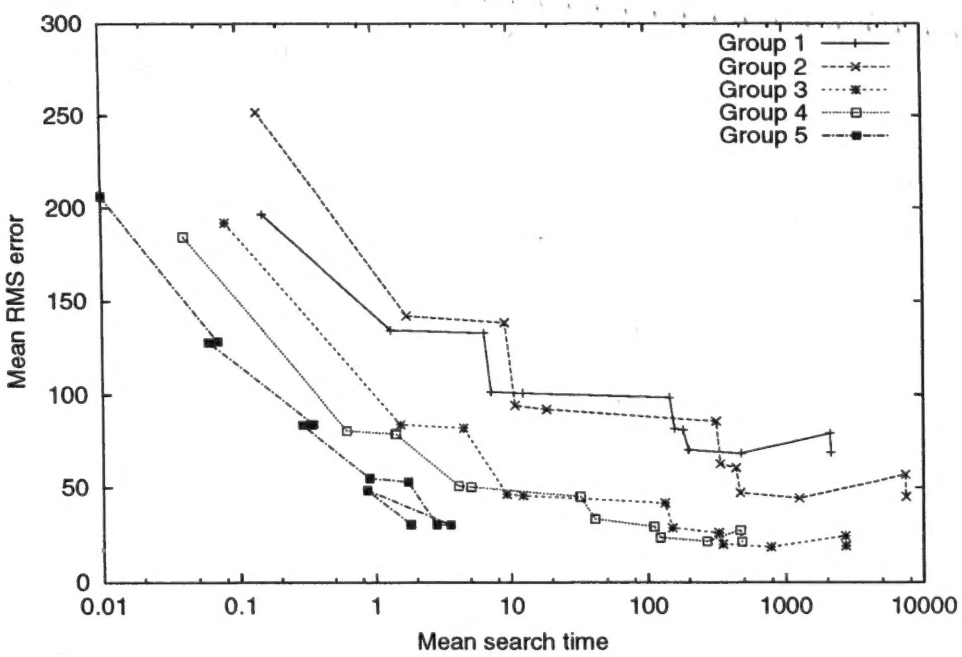


Figure 5.10: Graph of multi-level exhaustive search results.

The table subheadings T_{GA}, T_{aug}, and T_{srch} represent the time (in seconds) required for the genetic algorithm search, the time to augment a GA search, and the total search time of an augmented GA search (an initial GA followed by an exhaustive augmenting search), respectively; "Gen" indicates the number of generations for which the GA executed, and "Error" signifies the RMS matching error after the search.

The execution times indicated are for a version of the GA which features caching of LUP decompositions and least-squares solutions. Without caching, a "quick" GA search of depth 3 on the first tone of Group 1 (a bassoon A♯1) took 145 seconds, compared to just over 16 seconds with caching.

The GA search results are summarized in Figure 5.11, which compares the mean search times and RMS error levels from Tables A.4–A.7 with the multi-level exhaustive search results for Group 1. Each table of GA results is indicated by a separate line in the graph, since it is clear that a GA search by itself results in significantly higher error levels than a GA search augmented by a second-level exhaustive search.

The lone point to the right of the lower line labelled "Thorough GA" represents the results of a "2+2" augmented GA search, which is not included in the tables in Appendix A. The average RMS matching error of this search on Group 1 was 80.7 (SD 35.5); the average augmentation time was 299.7 seconds (SD 231.5), for a total average search time of 376.6 second (SD 288.6). These results are better than those of a "3+1" GA search (which yielded an average

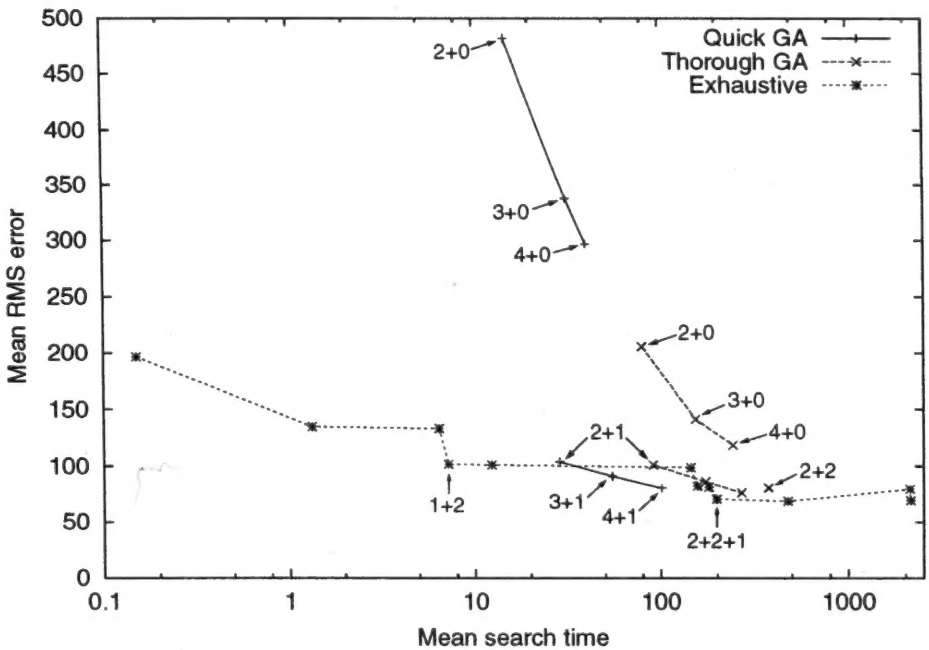


Figure 5.11: Graph of GA search results compared with multi-level exhaustive search for Group 1. Most GA data points are labelled, as are two significant points of multi-level exhaustive search results.

RMS error 86.1), but the search took more than twice the time; similarly, the GA "2+2" error level is slightly better than that of the match found by exhaustive "2+2" search (81.4), but again at over twice the cost. If five oscillators are to be used, the GA "4+1" search would be preferred, since it was both faster and better (RMS error of 76.1 in 271 seconds) than the GA "2+2" search, but the exhaustive "2+2+1" search was better still (70.5 error in 199 seconds).

In summary, a genetic algorithm may be useful as the initial stage of a multi-level search strategy, but only if caching of intermediate results is implemented; even then, some form of multi-level exhaustive search will likely yield better matches in less time than an augmented GA search of the same total depth.

5.5.3 Oscillator Assignment Optimization

The results of oscillator assignment optimization for Group 1 are presented in detail in Appendix B, and for the other groups, in the technical report [62]. Tables B.1–B.2 (pp. 161–162) show the results of 3-oscillator assignment optimization, Tables B.3–B.4 (pp. 163–164) present the 4-oscillator case, and Tables B.5–B.6 (pp. 165–166), 5-oscillator optimization. Each table presents

the results of optimizing the matches found by three different searches, indicating the optimization time (in seconds), the total time (search time plus optimization time, in seconds), and the RMS error for each tone in the relevant group for each type of initial search.

To prevent unreasonably large optimization graphs, a limit[8] of 15 was specified for the size of wavetable sets produced by overlapping in the 5-oscillator case, and the results in Tables B.5–B.6 reflect this choice. The implications of selecting this and other similar options are explored in Table 5.11 and in Tables B.7 and B.8 on pages 167 and 168 in Appendix B.

Each table compares the results of the default configuration—overlapping at a distance of two, with no limit on wavetable set sizes—with those of three restrictions on the construction of wavetable sets: size limits of 15 and 12, and restricting overlapping to a distance of one. Table 5.11 shows statistics on the direct action of these restrictions, listing the mean wavetable set size after overlapping of initial "4+0" matches of all tones in each group, the maximum and minimum mean set size for individual tones in each group, and the maximum wavetable set size constructed for any breakpoint of any tone in each group. Table B.7 compares the optimization time and RMS error after a 5-oscillator optimization of the "4+0" matches for each tone in Group 1 under the four configurations, and Table B.8 compares the numbers of vertices and edges in the optimization graph for each option.

It will be seen from the data in these tables that limiting wavetable set sizes to 15 is a modest restriction:

- Mean wavetable set sizes are reduced by less than 10% overall—a reduction of only about one wavetable in each set—and by no more than 15% for any individual tone.

- The size of the optimization graph, and thus optimization time, are reduced in the average case by about 15%, but no restriction is imposed on graph size in over half of the individual cases, and the largest graphs are reduced in size by over one-third.

- The mean RMS error of the final matches is increased by only about one-tenth of a percent.

A wavetable size limit of 12 and restricting overlapping to a distance of one increase the mean error by about 2% and 3%, respectively, but reduce optimization time to about one-quarter and one-eighth of unrestricted overlapping at distance 2. These options should be considered for use if speed is more important than the quality of final matches.

Some preliminary observations may be made based on the data in Tables B.1–B.6. For convenience, the examples offered in support of these observations will be derived from the first two tables in Appendix B (the 3-oscillator

[8]If a re-augmentation phase is triggered after the first phase of overlapping, the sizes of some of the final wavetable sets might exceed this limit.

Wavetable Set Size	Overlap Distance or WT Set Size Limit			
	Overlap=2	Limit=15	Limit=12	Overlap=1
Group 1				
Group Mean	11.7	11.5	9.9	8.8
Max. Mean	13.7	13.3	11.5	10.4
Min. Mean	9.1	9.1	8.6	7.3
Maximum	18	17	14	17
Group 2				
Group Mean	13.3	12.1	10.0	9.33
Max. Mean	15.7	13.4	10.9	10.6
Min. Mean	9.0	9.0	8.5	7.1
Maximum	23	16	16	16
Group 3				
Group Mean	14.4	12.5	10.2	9.8
Max. Mean	16.1	13.7	11.4	10.7
Min. Mean	9.5	9.5	8.8	8.1
Maximum	20	15	14	13
Group 4				
Group Mean	14.2	12.7	10.3	9.8
Max. Mean	15.8	13.3	10.7	10.5
Min. Mean	11.3	11.1	9.5	8.2
Maximum	19	16	16	14
Group 5				
Group Mean	9.2	9.2	9.2	7.6
Max. Mean	9.8	9.8	9.8	8.0
Min. Mean	7.5	7.5	7.5	6.8
Maximum	12	12	12	11
Overall				
Group Mean	13.3	12.1	10.1	9.4
Max. Mean	16.1	13.7	11.5	10.7
Min. Mean	7.5	7.5	7.5	6.8
Maximum	23	17	16	17

Table 5.11: Mean and maximum sizes of wavetable sets created by overlapping of "4+0" matches with limits on overlapping distance or wavetable set size.

matches for Group 1, pp. 161–162), but similar instances can be seen in the other result data as well.

- The best final matches do not necessarily result from optimizing the best initial matches. For example, the best final 3-oscillator match to the bass clarinet ("clb") G2 tone resulted from optimizing the initial "1+2" match, even though the "3+0" and "2+1" initial matches were better (see Tables A.1, A.2, and A.3).

- The best final matches for different tones may result from optimizing initial matches found by different types of searches. For example, optimization of the "3+0" match gave the best result for the piano G2 tone, the "2+1" match led to the best final match to the 'cello ("vlc") A♯2 tone, and the initial "1+2" match to the piano E2 yielded a significantly better final match than initial matches found by any other type of search.

- Initial matches of higher dimension tend to produce better final matches than lower-dimension initial matches, but do not always do so. For example, most of the 3-table optimized matches of Table B.1 are better than those of Table B.2, which were derived from initial 1- and 2-table matches. However, the optimized "2+0" and "1+1" matches to the horn A♯2 are better than those found beginning with 3-table initial matches, and all three of the lower-dimension matches (not only the "2+0" and "1+1," but also the "1+0" match) to the trombone C♯3 and both the bass ("vlb") A♯2 and C♯3 are better than those derived from any of the initial 3-table matches.

- In some cases, final matches of the same quality are found by optimizing any of the initial matches of a given size. For example, all three optimized matches in Table B.1 have the same RMS error (to one decimal place of accuracy) for each of the horn C♯3, the sax E2 and G2, the trombone E2, G2, and A♯2, and the bass G2. Furthermore, in some cases, such as the bass clarinet A♯2 and the 'cello E2, all six of the tested optimizations resulted in the same error level.

- Optimizations which take more time (due to constructing and traversing a larger graph) do not necessarily produce better final matches. For example, the best 3-oscillator matches to both the bass clarinet G2 and the horn E2 were found by the shortest of the six optimization phases for each tone. The best final matches to the piano A♯2 were found by both the shortest ("2+1") and the longest ("1+2") optimizations of the six tests of 3-oscillator matches.

The mean optimization time, total time, and RMS error of each search type for each group and number of oscillators are summarized for ease of comparison in Table 5.12. The lowest mean RMS error value is highlighted

for each group within each section of the table; if more than one search type yielded the same minimum error within a category, the value corresponding to the smallest optimization and search times is highlighted.

Figures 5.12–5.14 show the mean RMS error of each group as functions of mean optimization time and mean total time for 3-, 4-, and 5-oscillator matches, respectively; as in the graphs of search results, the time axis uses a logarithmic scale.

In most of the graphs, lines connect the data points, one line per group, such that each line traverses the points in the same order as they appear in the summary table, which lists the various search types within each section in the same order as in Table 5.10 (i.e., in increasing order of search time). Since, in general, optimization time is positively related to search time, the lines on these graphs proceed from left to right, for the most part. The first line segment in Figure 5.12, representing 3-oscillator optimization of matches found by an initial "1+0" search, is an exception; in this case, optimization time dominates search time, and a re-augmentation phase is triggered for every tone but one.

Mean optimization times of the 5-oscillator case are illustrated with points only in the upper graph of Figure 5.14 because the relationship between search time and optimization time does not hold in this case. In fact, there is no ordering of search types which would present the data points of each of the groups of this data set in increasing order, either by optimization time or by total time. The points of lowest error in each group are labelled for reference purposes.

Some general observations and recommendations follow from the data represented in the graphs of Figures 5.12–5.14 and listed in Table 5.12:

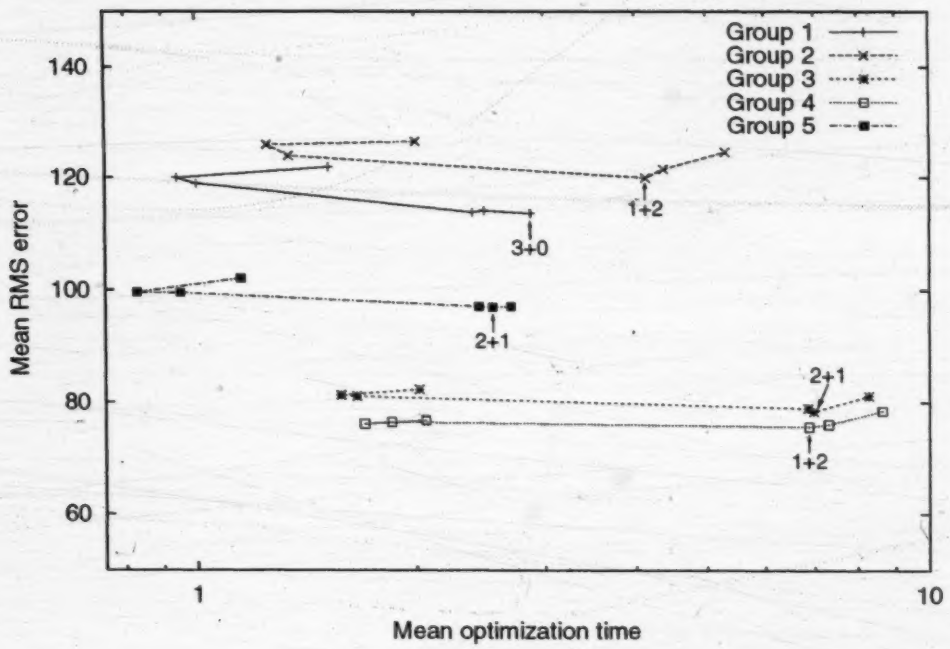
- Single-level exhaustive searches (especially "4+0" searches) tend not to be worth the computation time they require. Although the best 4-oscillator result for Group 5 was achieved by optimizing an initial depth-4 exhaustive search, the same process gave poorer results for Groups 2, 3, and 4, and in Group 1, a faster process—optimization of "3+1" matches—gave the same (best) result as the "4+0" optimization.

  This conclusion might also apply to the "3+0" search relative to 3-oscillator solutions, since the plot lines for Groups 2, 3, and 4 in Figure 5.12 show the same upward inflection at the right end as those for the same groups of the 4-oscillator case in Figure 5.13. If only 3-oscillator solutions are desired, depth-3 exhaustive searches should be eschewed in favor of "1+2" and "2+1" which lead to better or, in the case of Group 1, almost-as-good final matches in an order of magnitude less time. On the other hand, if 4- and 5-oscillator matches will also be tested, "3+0" searches will be well worth performing, since the results can be re-used in subsequent "3+1" and "3+2" searches and optimizations.

  The same logic does not necessarily extend to "4+0' searches. Although the results of such a search could be re-used for a "4+1" search preceding

Search Type	Group 1			Group 2			Group 3			Group 4			Group 5		
	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
3 Oscillators															
1+0	1.52	1.67	121.9	2.00	2.13	126.6	2.02	2.10	82.3	2.06	2.10	76.8	1.15	1.16	102.1
1+1	0.94	2.25	119.9	1.25	2.98	126.0	1.58	3.13	81.3	1.70	2.32	76.2	0.83	0.90	99.6
2+0	1.00	7.41	118.9	1.34	10.5	124.0	1.66	6.12	81.1	1.85	3.26	76.5	0.95	1.01	99.5
1+2	2.39	9.58	113.7	4.13	14.9	**120.0**	6.86	16.2	79.0	6.88	11.0	**75.7**	2.44	2.79	97.1
2+1	2.48	14.8	114.0	4.37	22.8	121.6	6.98	19.3	**78.4**	7.31	12.4	76.1	2.55	2.90	**97.0**
3+0	2.87	148	**113.5**	5.31	323	124.7	8.29	142	81.2	8.64	40.9	78.5	2.70	2.99	97.1
4 Oscillators															
1+2	13.3	20.5	97.8	24.3	35.1	93.3	42.8	52.2	53.2	42.2	46.3	54.7	10.2	10.6	72.0
2+1	13.2	25.5	97.6	26.2	44.6	92.5	44.3	56.6	53.0	46.9	52.0	54.3	11.1	11.4	71.9
3+0	15.6	160	96.9	32.7	350	91.1	54.6	189	53.3	59.3	91.6	51.3	12.5	12.8	65.9
3+1	46.5	203	**94.5**	119	458	**87.7**	187	338	**52.1**	186	227	49.2	23.9	24.8	65.8
2+2	54.4	236	95.1	122	563	91.0	197	524	53.0	195	306	**48.1**	25.4	27.2	67.5
4+0	66.8	2181	94.5	146	7595	91.8	234	2941	54.8	225	695	49.8	25.6	26.5	**65.1**
5 Oscillators															
3+1	220	377	83.5	399	738	69.9	563	715	**36.9**	628	669	36.3	78.4	79.3	47.5
2+2	253	434	83.2	453	894	70.0	617	945	37.3	659	770	**34.3**	87.2	88.9	46.3
2+2+1	340	540	81.9	394	866	**69.7**	487	839	39.0	537	660	35.6	161	163	**42.3**
3+2	358	835	81.8	386	1651	70.2	463	1246	40.4	549	818	36.8	176	180	42.3
4+0	284	2398	82.5	433	7883	70.3	577	3283	38.6	652	1123	34.8	87.6	88.5	47.0
4+1	351	2483	**81.5**	399	7881	70.5	471	3202	40.2	503	985	35.6	167	169	42.3

Table 5.12: Summary of 3-, 4-, and 5-oscillator optimization results for various initial search types.



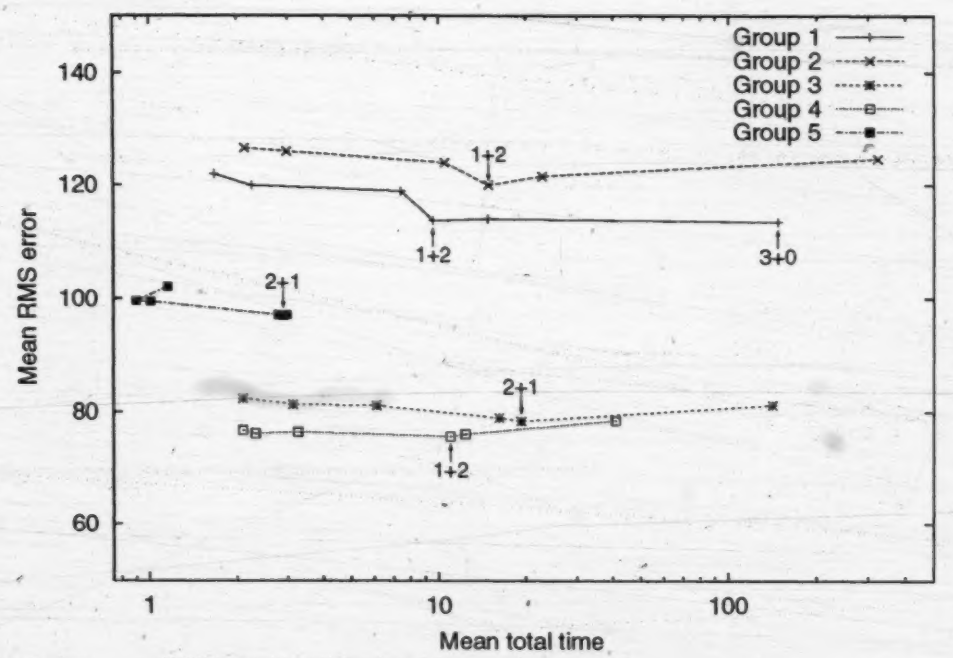


Figure 5.12: Graphs of 3-oscillator optimization results. Points of lowest error in each group are labelled.

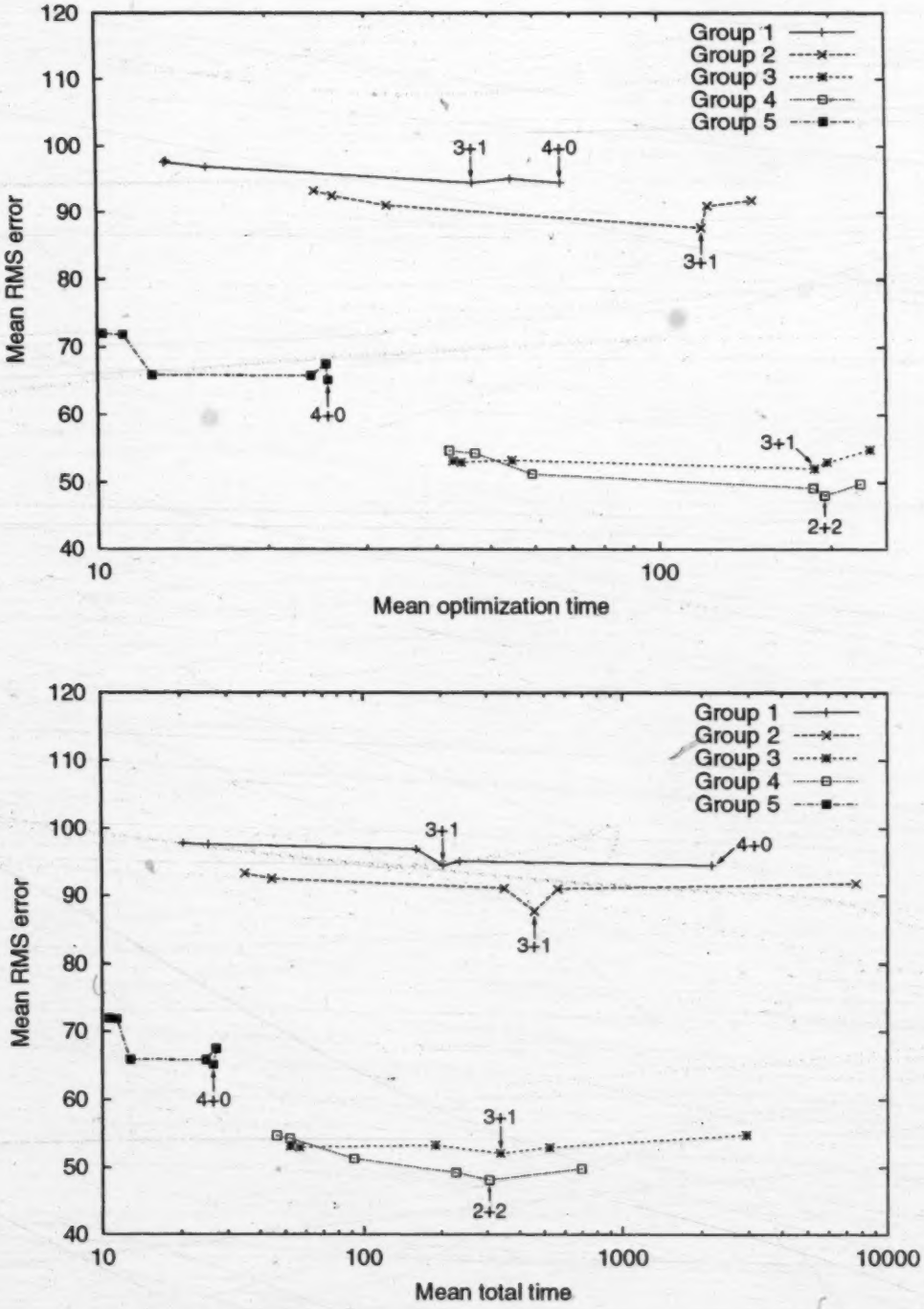
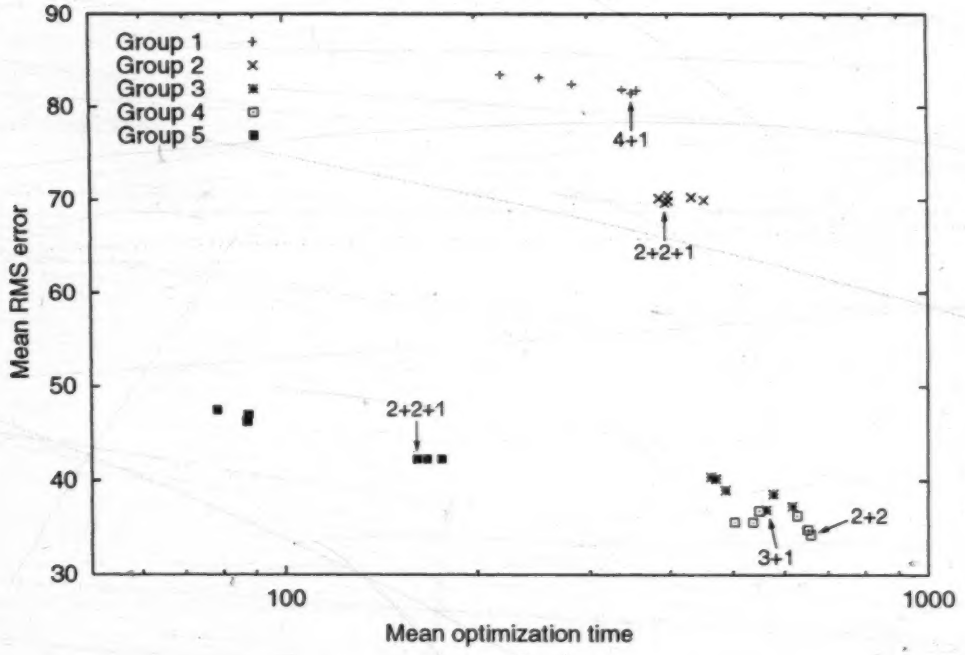


Figure 5.13: Graphs of 4-oscillator optimization results. Points of lowest error in each group are labelled.



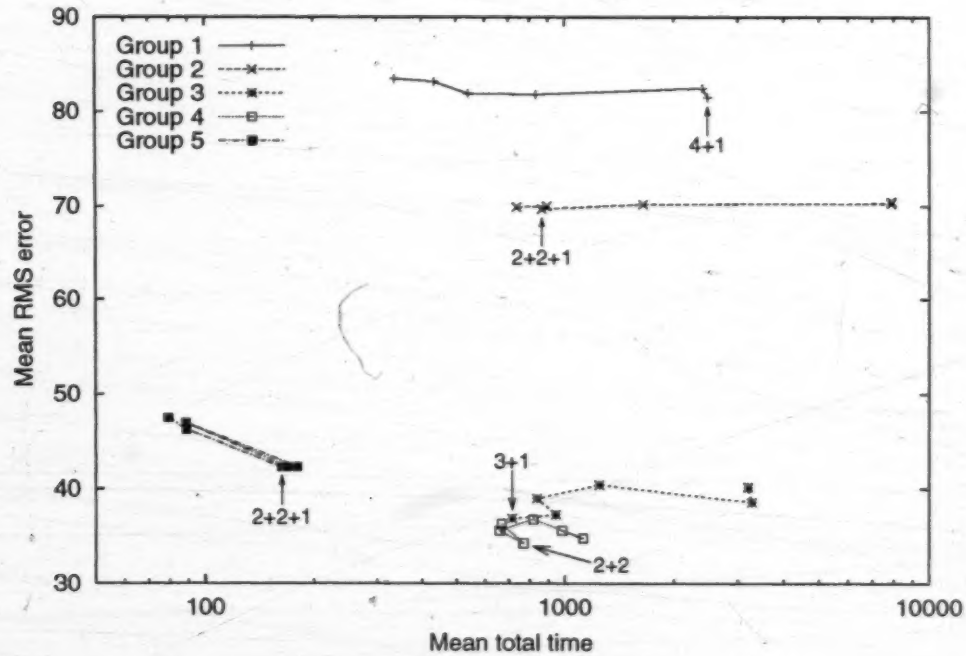


Figure 5.14: Graphs of 5-oscillator optimization results. Points of lowest error in each group are labelled.

a 5-oscillator optimization, the depth-4 exhaustive search is at least an order of magnitude more expensive than a depth-3 search. Furthermore, a "4+1" search is probably not worth executing, since results of equal or almost equal quality were achieved by optimizing initial "2+2+1" or "3+2" matches.

- In general, the best n-oscillator final matches are achieved by optimizing n-table initial matches. For example, none of the lowest mean RMS error levels are achieved in the 3-oscillator case by optimizing initial "1+0," "1+1," or "2+0" matches; similarly, initial "1+2," "2+1," and "3+0" searches lead to poorer 4-oscillator final matches than initial searches of depth 4.

  However, this observation does not apply to 5-oscillator matches: the best result for Group 3 is derived from an initial "3+1" search, and the best of Group 4 from a "2+2" search.

- No single search and optimization method gave the best results in all cases. A good strategy would be to optimize the matches found by two different medium-cost searches and use the better of the final two matches for each instrumental tone. Table 5.13 shows the mean and standard deviation RMS error achieved for each group by selecting the better final match of the 3-oscillator optimizations of initial "1+2" and "2+1" searches and of the 4-oscillator matches from initial "2+2" and "3+1" matches. In all 3- and 4-oscillator cases, the results due to this strategy are better, on average,[9] than the best result found by any individual search and optimization.

  For 5-oscillator matches, selecting the better of the final matches found by optimizing "2+2+1" and "3+2" matches resulted in a lower mean error level than the optimized "4+1" matches for Group 1, but did not equal the low mean error levels achieved by the "3+1" and "2+2" cases for Groups 3 and 4, respectively. However, selecting the best of three matches found by optimizing "3+1," "2+2," and "2+2+1" initial matches equalled or bettered all individual results in all groups. If "3+1" and "2+2" searches have already been executed as part of a 4-oscillator solution, this is a low-cost method of achieving excellent 5-oscillator matches, since the only additional exhaustive search which is required is the third-level search of depth 1 to augment the "2+2" initial matches, creating "2+2+1" initial matches.

[9] Better results can be achieved for individual tones by optimizing other initial searches, but the suggested strategy produces lower *mean* RMS error levels than the optimization of any single multi-level exhaustive search.

Error	Group 1	Group 2	Group 3	Group 4	Group 5
Best of 1+2 and 2+1 Matches for 3 Oscillators					
Mean	113.4	119.0	78.0	75.4	97.0
Std. Dev.	43.2	76.0	58.9	88.5	134.6
Best of 2+2 and 3+1 Matches for 4 Oscillators					
Mean	93.9	86.7	51.6	46.0	63.5
Std. Dev.	38.1	60.7	44.0	60.8	85.9
Best of 2+2+1 and 3+2 Matches for 5 Oscillators					
Mean	80.9	68.4	38.5	35.0	42.2
Std. Dev.	35.4	52.1	35.6	51.5	55.7
Best of 2+2, 3+1, and 2+2+1 Matches for 5 Oscillators					
Mean	81.3	67.4	36.3	33.2	42.3
Std. Dev.	35.7	50.8	33.5	50.6	55.7

Table 5.13: Mean matching error levels achieved by selecting the best match to each tone from among the optimizations of multiple different initial search strategies.

5.5.4 Optimization of Matches Found with a GA Search

Tables B.9 and B.10 (pp. 169 and 170 in Appendix B) present the results of optimizing the initial "2+1," "3+1," and "4+1" matches found by a first-level genetic algorithm search augmented with a depth-1 exhaustive search for 3-, 4-, and 5-oscillator assignments, respectively. As with the 5-oscillator final matches in the previous section, wavetable set sizes were limited to 15 for the 5-oscillator "4+1" match.

The graphs in Figure 5.15 gather the curves depicting the 3-, 4-, and 5-oscillator optimizations of multi-level exhaustive searches for Group 1 from Figures 5.12, 5.13, and 5.14 and plot the data points for the mean results (optimization time and total time versus error) of the optimized "2+1," "3+1," and "4+1" GA matches, both the thorough and quick versions, from Tables B.9 and B.10.

The results reported in the table and illustrated in the graphs were the best of the optimizations of initial matches found by a genetic algorithm. One other tested optimization of a GA match was of interest: the 4-oscillator optimization of a "2+2" GA match yielded a mean RMS error of 94.9 (SD 38.8), which is lower than the "3+1" GA match (96.0) and slightly better that of the optimization for 4 oscillators of the exhaustive "2+2" search (95.1), but not as good as the exhaustive "3+1" 4-oscillator optimization (94.5). With a mean optimization time of 57.6 seconds (SD 38.6) and a mean total time of 434 seconds (SD 320), the "2+2" GA match is not time-competitive with either the "2+2" or "3+1" exhaustive search optimizations.

It is clear that there are 4- and 5-oscillator exhaustive search optimizations which give significantly better results than the optimizations of both "quick" and "thorough" GA matches in less time. In the case of 3-oscillator

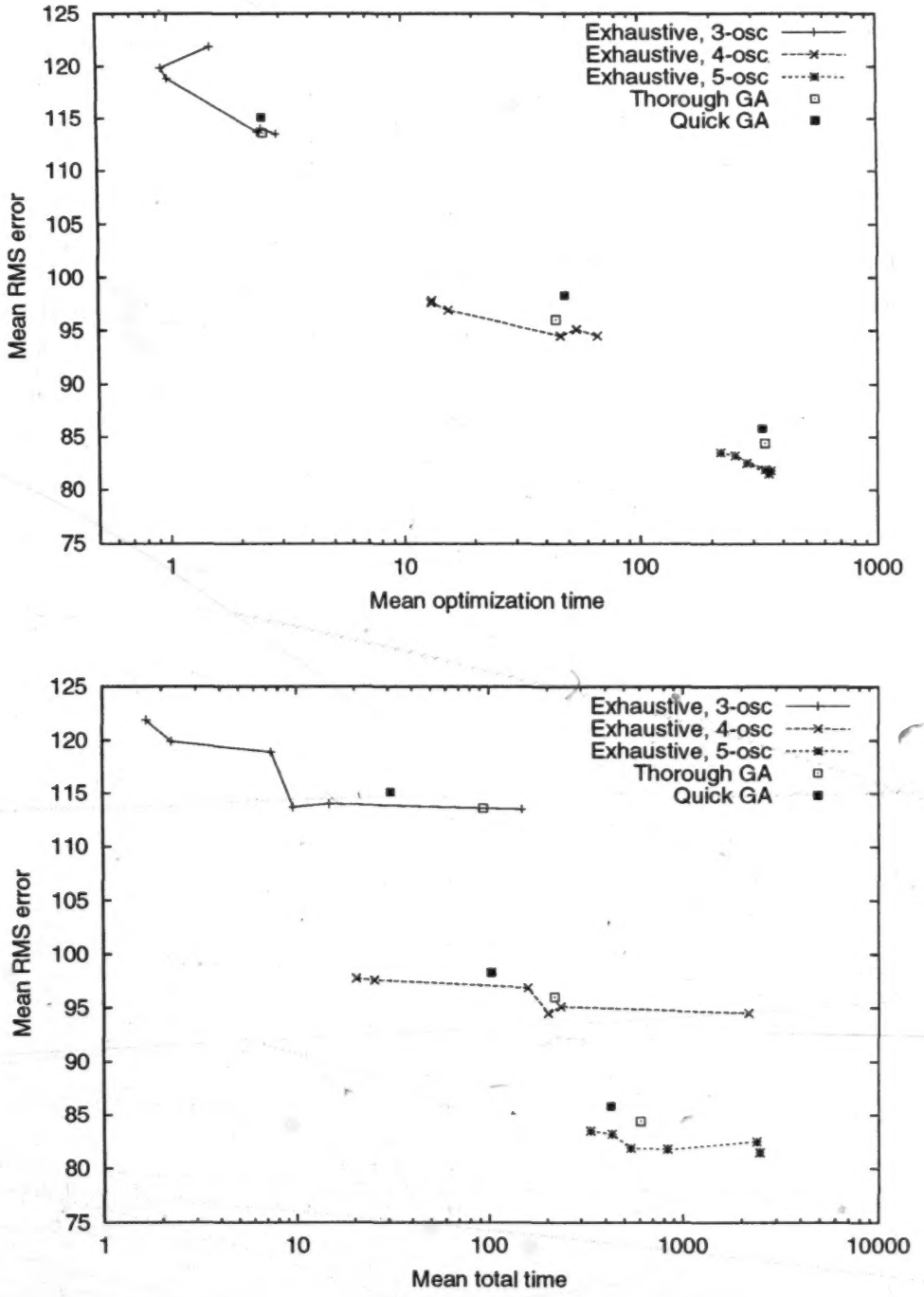


Figure 5.15: Graph of GA optimization results compared with 3-, 4-, and 5-oscillator optimizations of multi-level exhaustive searches for Group 1.

optimizations, the GA matches were more competitive, but given the relatively low computational demands of 3-oscillator optimizations, the strategy recommended in the previous section would be preferred.

5.5.5 Comparison with Horner's Method

As discussed in §2.4.9, Horner [42] used a simple form of oscillator assignment for his experimentation with multiple wavetable interpolation: a match of size $N_{osc} - 1$ was found by exhaustive search for the breakpoint spectrum with the largest RMS amplitude, and the other breakpoint spectra were matched by working forward and backward from the peak breakpoint, changing at most one wavetable from one breakpoint to the next by exhaustive search. This method might be called *constrained matching*, since all matches are constrained to consist of $N_{osc} - 1$ wavetables.

Table B.11 in Appendix B presents the results of implementing and testing Horner's constrained matching on the instrumental tones of Group 1, using the same wavetable bank for this group of tones as was selected for testing the method of optimized multiple wavetable interpolation proposed here. Results for the other groups are given in the technical report [62].

It should be clearly noted that this comparison with Horner's method is restricted to the breakpoint spectrum matching and oscillator assignment components only. Horner did not use large, general-purpose wavetable banks, but used a genetic algorithm to select small sets of basis spectra (two to ten wavetables in size) which were particular to each instrument being matched.

Figure 5.16 compares the results of Horner's constrained matching method with those of optimized matching using 3, 4, and 5 oscillators for the tones of Group 1.

While Horner's constrained matching method is faster than any of the types of multi-level exhaustive search optimization for a given number of oscillators, the error levels produced by Horner's method are significantly higher than those of the optimized matches, and are closer to those achieved by optimization with one fewer oscillators.

Figure 5.17 shows the oscillator assignments derived using Horner's constrained matching method with four oscillators on a French horn E2 tone. When compared with Figure 3.14 (p. 59), which shows the oscillator assignments found by optimization of matches to the same horn tone, the figure illustrates why the results of Horner's method approach those of optimized matching with one fewer oscillators. The dashes in the four main columns of the figure indicate the points at which an oscillator is required for the fade-out of one wavetable and the fade-in of another; occurrences of wavetable index zero indicate breakpoints at which an oscillator is idle between that breakpoint and the next. The peak amplitude occurs at breakpoint 6, so the set of wavetables used at that point constitute the best match to that spectrum that could be found by an exhaustive search of depth 3. Replacing any single wavetable of that set by another wavetable failed to produce a better match at

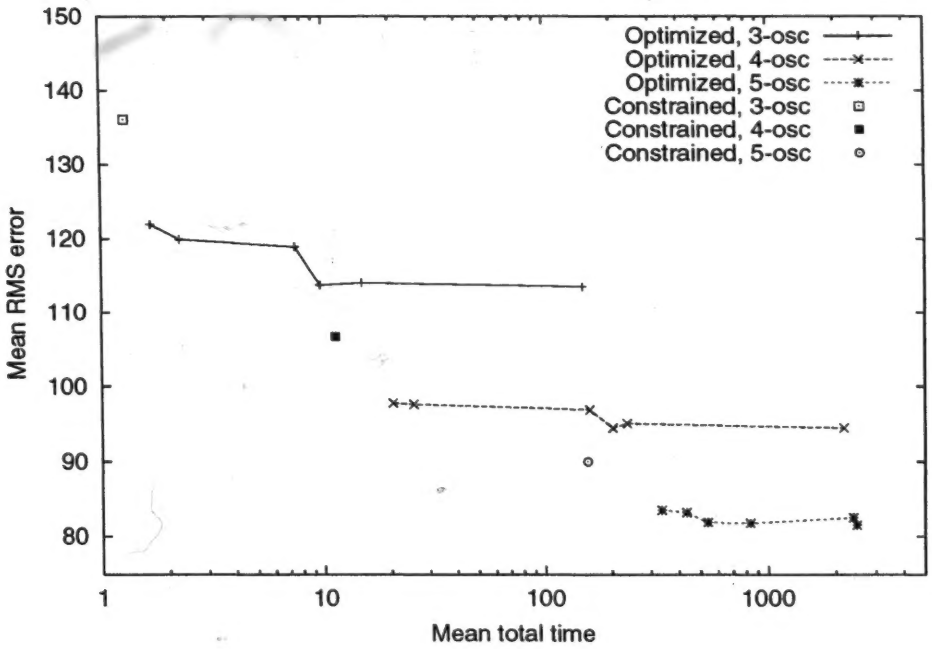


Figure 5.16: Comparison of Horner's constrained matching with optimized multi-level exhaustive search results for Group 1.

breakpoint 5, so the same set of wavetables was used there, leaving the second oscillator unassigned. Similarly, the set {10, 15, 37} was the best match that could be found for the spectra at breakpoints 7 to 14, subject to the condition that only one wavetable could be changed at a time; this left the first oscillator unused until breakpoint 14, at which point it started fading in wavetable 6, which it had previously faded out between breakpoints 6 and 7. Additional examples of fading out a particular wavetable, only to fade it back in, may be seen at breakpoints 15, 16, and 20. If it is assumed that wavetable 6 could have contributed to the quality of the matches at breakpoints 7–14 and 16, and that wavetables 4, 10 and 37 might have been similarly useful at breakpoints 2, 5, 15, 17–18, and 20, then there are at least 15 points—five-eighths of the breakpoints—at which an unused oscillator could have been gainfully assigned to achieve higher quality matches.

If oscillators are a cheaper and more plentiful resource than computation time, or if the speed with which results are achieved is more important than the quality of the results, then Horner's method is to be recommended. However, since the spectral matching procedure need be performed only once for each tone, a higher quality result using fewer oscillators would typically be desired, and the time required to achieve that goal would be of secondary importance. It is for this reason that the proposal of a method for selecting optimal matches which fully utilize the available oscillators is a significant contribution.

Break-	Oscillator			
point	1	2	3	4
1:	6	10	15	—
2:	6	0	15	9
3:	6	—	15	9
4:	6	4	15	—
5:	6	0	15	37
6:	6	—	15	37
7:	0	10	15	37
8:	0	10	15	37
9:	0	10	15	37
10:	0	10	15	37
11:	0	10	15	37
12:	0	10	15	37
13:	0	10	15	37
14:	—	10	15	37
15:	6	—	15	37
16:	—	10	15	37
17:	6	0	15	37
18:	6	0	15	37
19:	6	—	15	37
20:	6	29	15	—
21:	6	—	15	37
22:	—	4	15	37
23:	40	—	15	37
24:	0	8	15	37

Figure 5.17: Oscillator assignment resulting from Horner's method for a four-oscillator match of a French horn E2.

Number of Breakpoints	Size		
	3 Osc.	4 Osc.	5 Osc.
24	848	1056	1264
48	1616	2016	2416
85	2800	3496	4192
122	3984	4976	5968
179	5808	7256	8704
Total	304928	380368	455808

Table 5.14: Sizes (in bytes) of wavetable envelope files (stored control streams) for tones with various numbers of breakpoints using 3, 4, or 5 oscillators.

5.5.6 Data Reduction

The data reduction achieved through multiple wavetable interpolation analysis is significant: a reduction of two orders of magnitude.

The stored size of an original digital waveform is determined by its duration, the sampling rate, and the data representation of the samples. A CD-quality waveform stored in a typical WAV file format (16-bit signed integer samples) requires 44100 × 2 bytes per second of sound, plus a 44-byte header. The WAV files of the tones selected for this study (see Table 5.1, p. 94) range in size from 88168 bytes (the glockenspiel G5) to 724450 bytes (the trumpet C♯4), with an average size of 294 kilobytes, and occupy a total of 58.2 megabytes.

As shown in Table 5.9 (p. 116), all five wavetable banks occupy only 57.4 kilobytes.

The oscillator assignment control stream for each analyzed and matched tone consists of a single-precision floating-point value (4 bytes) for the time index of each breakpoint, another for the pitch differential at that breakpoint, and, for each oscillator, an unsigned integer (4 bytes) for the index of a wavetable in the bank and a floating-point weighting (amplitude coefficient) for that wavetable. For a 4-oscillator control stream, this totals only 40 bytes per breakpoint. For convenience, a control stream record is also stored for the external breakpoints, and the control stream for each tone is stored in a file with a 16-byte identifying header. Table 5.14 indicates the sizes of these wavetable envelope files for the typical cases (24 and 48 internal breakpoints) and for some of the other cases, in which various numbers of breakpoints were used. The total size of the 198 wavetable envelope files for each of the 3-, 4-, and 5-oscillator cases is also indicated.

The total space occupied by the 198 5-oscillator wavetable envelope files plus the space required for the five wavetable banks is 513 kilobytes, which is only 0.88% of the space used by the 198 original CD-quality WAV files.

5.6 Synthesis Results

Three-, four-, and five-oscillator optimized matches were selected for each test tone according to the method recommended at the end of §5.5.3 and the tones were resynthesized using `Csound`. An informal listening test was conducted by the author to compare each of the synthesized tones with its respective original instrumental tone, evaluating the fidelity of each according to the scale previously shown in Table 2.1 (p. 25).

Many of the synthesized tones were deemed to be perceptually indistinguishable from their original correspondents, especially the lower bassoon tones; the mid- to upper-range clarinet tones; the bass clarinet tones from Group 2; most of the English horn, French horn, and oboe examples; almost all of the saxophone tones in Groups 2, 3, and 4; most of the upper trumpet notes; and almost all of the trombone selections. In most cases, all three variants—the 3-, 4-, and 5-oscillator syntheses—were deemed to be of equal quality, but in some cases, only the higher-order forms were perceptually indistinguishable from their sources. In these latter cases, the audible differences in the 3-oscillator resyntheses were typically due to rapid timbre changes in the attack or release portions which the 3-oscillator solution either failed to fully capture or exaggerated so that the changes were more audible in the reconstruction than in the original.

Most of the bass clarinet, English and French horn, oboe, sax, trumpet, and trombone tones that were not deemed to be perceptually indistinguishable were thought to be musically indistinguishable from their originals (category 2 on Moore's scale). About two-fifths of the string tones were similarly categorized, especially the middle to upper viola tones, the lower bass tones, and the mid- to upper-range violin tones. The perceived differences were primarily due to one of two causes: noise or transients in the attack portion (and occasionally in the release) of the original tones, and exaggeration of frequency changes or timbral brightening in the release. For example, the trumpet G3 features a rapid timbre change in the attack which is quite clearly the result of the onset of vibration of the player's lips in the mouthpiece, and an aggressive pulling and releasing of the string by the bow gives the 'cello C♯2 an exciting start; neither of these features is well captured in the resyntheses, since multiple wavetable synthesis is not amenable to the reconstruction of noise. The frequency of the 'cello E2 tone is heard to rise as the 'cellist lifts the bow from the string with extra energy at the end of the tone, and this frequency change is exaggerated in the resynthesis, no doubt due to the simplicity of the formula used to calculate a single weighted average frequency differential for all partials. The timbre brightens at the end of the string bass C♯2 due to a similar use of the bow, and this change is also exaggerated by the matching, probably due to the allocation of too few breakpoints to the lower-amplitude release section of the tone.

Some tones were heard as musically acceptable: they were audibly different from the originals, but could acceptably be substituted for the originals

in a musical context. In this category were the upper bassoon tones; the lower clarinet, saxophone, and bass clarinet pitches; most of the flute tones; a few horn, piano, trumpet, and violin pitches; and most of the viola, bass, and 'cello tones. In general, the synthesized tones sounded cleaner than their natural counterparts, which contained significant noise components. For the wind-generated tones, the noise was often due to the flow of air: in the bassoon C♯4, the sound is that of excess air escaping the instrument, and in the flute tones, the stream of air being blown across the embouchure[10] hole is especially audible. For the strings, the noise is that of the rosined[11] hairs of the bow scraping on the string. Both types of noise are given prominence in the source tones by the close placement of the microphone during recording; the synthesized variants sound more like their instrumental sources heard at a greater distance. A lip buzz is clearly audible in the original trumpet A♯3 tone which is not reproduced in its synthesized counterpart. In several of the piano tones (especially C♯5 and G5), the mechanical noises of the action hitting the hammer and lifting of the damper are clearly heard in the recordings; these noises are not reproduced on resynthesis.

As expected (see §5.2), the signals synthesized from matches to the glockenspiel and piano tones were musically different from their sources, as were a few other tones, including several flute and 'cello tones and one violin tone. The recorded piano tones are characterized by a directness in the attack, in which the stiffness of the strings can clearly be heard, which the resyntheses do not capture; especially in the lower synthesized piano tones, a buzz gives the attack portion an artificial character. In most of the glockenspiel tones, some of the upper piano tones, and the 'cello C♯3, the difference is due to pronounced or exaggerated frequency changes, again no doubt due to the calculation of a weighted average frequency differential. In the case of the glockenspiel and the piano, it is clear that the frequency centroid at the onset of the tone is strongly influenced by percussive attack transients, then shifts downward as the transients decay and the quieter sustained partials predominate in the calculation of the weighted average. When the frequency differential ramps were replaced by constant zeros and the tones were resynthesized, the results were musically acceptable, although the synthesized glockenspiel was somewhat less ringing than the original, and the thumping sound of the piano action was not reproduced.

[10] Mouthpiece.

[11] Rosin, a substance derived from pine resin, is rubbed on the hair of the bows of stringed instruments, as on the shoes of ballet dancers and the hands of gymnasts, to keep them from slipping.

Chapter 6

Conclusions and Suggestions for Further Research

The goals of this research effort were, in essence, the same as those of many research projects: *generality* and *optimality*. Two questions were central to the research: "Is it possible to generalize the techniques of multiple wavetable analysis/synthesis so that tones of many different instruments may be analyzed, matched, and resynthesized with reference to a common set of basis spectra?" and "Is it possible to perform spectral matching and oscillator assignment in a globally optimal way, rather than with ad hoc techniques?" Both these questions have been answered in the affirmative.

The generalization of multiple wavetable matching can be thought of as the use of a horizontal rather than a vertical grouping of tones. In previous studies which used a common set of basis spectra to match multiple tones, the tones were those of different pitches being played by the same instrument [41] or by instruments which had been determined to have similar timbres [89]. This might be characterized as a vertical grouping of tones, since the tones ranged from low to high in pitch, but did not range across different instruments or groups of instruments. These studies avoided the issue that basis spectra cannot be arbitrarily shifted in frequency (due to the limit imposed by the Nyquist frequency) by using band-limited spectra (i.e., by retaining only a limited number of harmonics in the basis spectra); this approach would seem to obviate the need for multiple wavetable synthesis since, if one is willing to forego the rich spectra created by many partials, one might as well use sinusoidal additive synthesis for the few harmonics that remain. In the current study, a horizontal grouping of tones was used instead—grouping together all the tones within a narrow pitch range from across all the instruments being considered—in order to address directly the Nyquist limit while generalizing the multiple wavetable technique across many different instruments.

The three-stage optimization method proposed in this thesis—selecting an initial match to each breakpoint spectrum, overlapping the matches of adjacent breakpoints to form wavetable sets, then optimizing oscillator assignments by constructing a vertex-weighted directed acyclic graph and executing the single-

source acyclic weighted shortest path algorithm on it—avoids the two ad hoc restrictions of Horner's method: that the match to each breakpoint spectrum be constrained to a size one less than the number of oscillators to be used in synthesis, and that the matching process begin with the peak-amplitude breakpoint and work outward. It has been demonstrated that the optimization method yields significantly improved matches compared to Horner's constrained matching method.

Three other techniques were proposed and tested in this research:

- A new breakpoint selection algorithm based on segment merging was successfully used to construct piecewise linear approximations of spectral envelopes, and was found to be advantageous due to its speed of execution, its selection of timbral extrema, and its predictability as a *stack algorithm*: the set of breakpoints selected for an n-breakpoint PLA is a subset of those in a PLA having $n+1$ breakpoints.

- A new method of selecting basis spectra was successfully used to construct the wavetable banks, which are central to optimized multiple wavetable interpolation: the breakpoint spectra from all of the tones produced by the various different instruments which are within the pitch range of a particular horizontal group are submitted to a clustering algorithm, and the centroids of the clusters are used as the basis of the wavetable bank for that group. It was found, as anticipated, that spectra from different segments of different instrumental tones are frequently clustered together and that wavetable bank spectra which originated in the tones of one instrument are re-used in the synthesis of tones of other instruments. As a result of this generalized re-use of basis spectra, the wavetable banks consisted, on average, of only 1.2 wavetables per tone.

- A proposed multi-level search strategy proved to be very successful for finding initial matches to breakpoint spectra for the first stage of the optimization algorithm. In effect, this strategy prunes the search tree by restricting its search at level n to subtrees of those nodes which were found to be the best m-table matches to breakpoint spectra at level $n-1$ of the search, where m is, in the general case, less than the number of wavetables desired in matches to be submitted to the next stage of the optimizer. Two different search algorithms—exhaustive search and a genetic algorithm search—were tested for use at the first level of the multi-level search; exhaustive searches of the depths used in this research (typically 2 or 3, and not more than 4) were found to produce matches of equal or better quality than those generated by the genetic algorithm in approximately the same time. Exhaustive search of limited depth was always used at levels above 1.

The success of these techniques and the wavetable matching optimization algorithm was confirmed by an informal listening test.

The results of this research suggest several opportunities for further research:

- There is a need for further experimentation with parameters to the breakpoint algorithm, including the testing of alternative error measures, to improve the capture of salient details of the attack and release segments of tones. Attack weighting was used to improve the modeling of certain flute tones, but listening tests indicated that attack weighting may have improved the accuracy of the modeling of other tones as well. The selection of the appropriate weight to use and the duration over which it is applied is currently accomplished by experimentation and hand tuning, which is a time-consuming and ad hoc process. A better solution might be to use an adaptive error measure which, like Horner's relative error measure (see Equation 4.3), gives greater weight to the lower-amplitude parts of the tone but, unlike the relative error measure, does not overemphasize the portions with near-zero amplitude.

- A more sophisticated method of calculating a common frequency differential for all harmonics is clearly necessary, since the use of a simple weighted average does not distinguish between the contribution of strong early transients and that of sustained or more slowly decaying partials. Alternatively, a smoothing function might suffice to avoid the exaggerated frequency changes which were heard in several resynthesized tones, especially if the function applied greater smoothing to the frequency envelope in the lower-amplitude portions of the tone.

- Experimentation is required to determine if calculating a separate PLA for frequency differentials would allow the use of fewer breakpoints to model the spectral envelopes of tones with vibrato. Alternatively, the frequency differential at each frame could be included as one of the values used by the breakpoint selection error measure; it could be weighted more heavily than the partial amplitudes to ensure that the frequency maxima and minima of a vibrato are appropriately captured by the PLA.

- A promising avenue of research would be the addition of a stochastic component like that used in Spectral Modeling Synthesis (see §2.4.4) to the optimized multiple wavetable interpolation model in order to model the noise which is so clearly present in many instrumental tones. The inherently harmonic nature of multiple wavetable synthesis limits its ability to lend realism to the resynthesis of tones which include the scrapes, thumps, chiffs, and air flow noises of actual instruments but which are otherwise harmonic in structure.

Optimized multiple wavetable interpolation analysis/synthesis has been shown to be a useful model for the compact representation and rapid synthesis of a wide variety of instrumental tones. Using only a small collection

of data—the wavetable banks—and a sparse control stream—for each breakpoint, a time index, a frequency differential, and a wavetable index and weight for each oscillator—the method enables the synthesis of dynamically changing harmonic spectra with only a few oscillators. The technique of grouping tones by pitch range allows the use of different numbers of partials for tones of different pitch without creating synthesis artefacts due to exceeding the Nyquist frequency.

The method would be very suitable for incorporation in a hardware synthesizer, where it would offer both high fidelity and low cost. The model is also useful to electronic music composers in a software synthesis context, since its control stream is quite intuitive due to the use of common breakpoints and spectral ramps. The introduction of an oscillator assignment optimizer provides assurance to hardware designers and composers alike that the resources available, whether hardware oscillators or computational cycles, are used as fully as possible during synthesis.

Bibliography

[1] BACKUS, J. *The Acoustical Foundations of Music*, 2 ed. Norton, New York, 1977.

[2] BEAUCHAMP, J. W. A computer system for time-variant harmonic analysis and synthesis of musical tones. In *Music by Computers*, H. V. Foerster and J. W. Beauchamp, Eds. Wiley, 1969, pp. 19–62.

[3] BEAUCHAMP, J. W. Analysis and synthesis of cornet tones using nonlinear interharmonic relationships. *Journal of the Audio Engineering Society 23*, 10 (Dec. 1975), 778–795.

[4] BEAUCHAMP, J. W. Synthesis by spectral amplitude and "brightness" matching of analyzed musical instrument tones. *Journal of the Audio Engineering Society 30*, 6 (June 1982), 396–406.

[5] BEAUCHAMP, J. W. Unix workstation software for analysis, graphics, modifications, and synthesis of musical sounds. In *Ninety-fourth Convention of the Audio Engineering Society* (Berlin, 1993), Audio Engineering Society, New York. Preprint 3479.

[6] BEAUCHAMP, J. W. Analysis and resynthesis of percussion sounds: Two methods compared. In *Proceedings of the International Computer Music Conference* (1999), International Computer Music Association, pp. 347–350.

[7] BEAUCHAMP, J. W., AND HORNER, A. Extended nonlinear waveshaping analysis/synthesis technique. In *Proceedings of the International Computer Music Conference* (1992), International Computer Music Association, pp. 2–5.

[8] BERGER, K. W. Some factors in the recognition of timbre. *Journal of the Acoustical Society of America 36*, 10 (1964), 1888–1891.

[9] BLACKHAM, E. D. The physics of the piano. *Scientific American 213* (Dec. 1965), 88–97.

[10] CANN, R. An analysis/synthesis tutorial. In Roads and Strawn [81], ch. 9, pp. 114–144. Originally printed in *Computer Music Journal* 3(3):6–11, 1979; 3(4):9–13, 1979; and 4(1):36–42, 1980.

[11] CHAMBERLIN, H. Advanced real-time music synthesis techniques. *BYTE 5*, 4 (Apr. 1980), 70–94, 180–196.

[12] CHAPMAN, N., AND CHAPMAN, J. *Digital Multimedia.* Wiley, Chichester, 2000.

[13] CHARBONNEAU, G. R. Timbre and the perceptual effects of three types of data reduction. *Computer Music Journal 5*, 2 (1981), 10–19.

[14] CHEESEMAN, P., AND STUTZ, J. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. AAAI Press, Menlo Park, CA, 1995.

[15] CHOWNING, J. Forward: New music and science. In *The Computer Music Tutorial* [80].

[16] CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. R. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[17] DANNENBERG, R. B. Interpolation error in waveform table lookup. In *Proceedings of the International Computer Music Conference* (1998), International Computer Music Association, pp. 240–243.

[18] DEPALLE, P., GARCIA, G., AND RODET, X. Tracking of partials for additive sound synthesis using hidden Markov models. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (Minneapolis, 1993), pp. 225–228.

[19] DEUTSCH, D., Ed. *The Psychology of Music.* Academic, New York, 1982.

[20] DODGE, C., AND JERSE, T. A. *Computer Music: Synthesis, Composition, and Performance.* Schirmer Books, New York, 1985.

[21] DOLSON, M. The phase vocoder: A tutorial. *Computer Music Journal 10*, 4 (1986), 14–27.

[22] DOLSON, M. Fourier-transform-based timbral manipulations. In Mathews and Pierce [58], ch. 9, pp. 105–112.

[23] DUDLEY, H. The vocoder. *Bell Laboratories Record 17* (1939), 122–126.

[24] FEITEN, B., AND GÜNZEL, S. Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal 18*, 3 (Fall 1994), 53–65.

[25] FITZ, K., AND HAKEN, L. Bandwidth-enhanced sinusoidal modeling in lemur. In *Proceedings of the International Computer Music Conference* (Banff, 1995), International Computer Music Association, pp. 154–157.

[26] FITZ, K., HAKEN, L., AND CHRISTENSEN, P. A new algorithm for bandwidth association in bandwidth-enhanced additive sound modeling. In *Proceedings of the International Computer Music Conference* (Berlin, 2000), International Computer Music Association, pp. 384–387.

[27] FITZ, K., HAKEN, L., AND CHRISTENSEN, P. Transient preservation under transformation in an additive sound model. In *Proceedings of the International Computer Music Conference* (Berlin, 2000), International Computer Music Association, pp. 392–395.

[28] FITZ, K., WALKER, W., AND HAKEN, L. Extending the McAulay-Quatieri analysis for synthesis with a limited number of oscillators. In *Proceedings of the International Computer Music Conference* (1992), International Computer Music Association, pp. 381–382.

[29] FLANAGAN, J. L., AND GOLDEN, R. M. Phase vocoder. *Bell System Technical Journal 45* (1966), 1493–1509.

[30] FLETCHER, H., BLACKHAM, E. D., AND GEERTSEN, O. N. Quality of violin, viola, 'cello, and bass-viol tones. I. *Journal of the Acoustical Society of America 37*, 5 (May 1965), 851–863.

[31] FLETCHER, H., AND SANDERS, L. C. Quality of violin vibrato tones. *Journal of the Acoustical Society of America 41*, 6 (1967), 1534–1544.

[32] FLETCHER, N. H., AND ROSSING, T. D. *The Physics of Musical Instruments.* Springer, New York, 1991.

[33] FREEDMAN, M. D. Analysis of musical instrument tones. *Journal of the Acoustical Society of America 41*, 4 (1967), 793–806.

[34] GABOR, D. Acoustical quanta and the theory of hearing. *Nature 159*, 4044 (1947), 591–594.

[35] GEORGE, E. B., AND SMITH, M. J. T. Analysis-by-synthesis/overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones. *Journal of the Audio Engineering Society 40*, 6 (June 1992), 497–516.

[36] GOODRICH, M. T., AND TAMASSIA, R. *Data Structures and Algorithms in Java.* Wiley, New York, 1998.

[37] GORDON, J. W., AND STRAWN, J. An introduction to the phase vocoder. In Strawn [91], ch. 5, pp. 221–270.

[38] GREY, J. M. *An Exploration of Musical Timbre.* PhD thesis, Stanford University, 1975.

[39] GREY, J. M., AND GORDON, J. W. Perceptual effects of spectral modifications on musical timbres. *Journal of the Acoustical Society of America 63*, 5 (May 1978), 1493–1500.

[40] GREY, J. M., AND MOORER, J. A. Perceptual evaluations of synthesized musical instrument tones. *Journal of the Acoustical Society of America 62*, 2 (Aug. 1977), 454–462.

[41] HORNER, A. *Spectral Matching of Musical Instrument Tones.* PhD thesis, University of Illinois at Urbana-Champaign, 1993.

[42] HORNER, A. Computation and memory tradeoffs with multiple wavetable interpolation. *Journal of the Audio Engineering Society 44*, 6 (June 1996), 481–496.

[43] HORNER, A., AND BEAUCHAMP, J. Piecewise-linear approximation of additive synthesis envelopes: A comparison of various methods. *Computer Music Journal 20*, 2 (Summer 1996), 72–95.

[44] HORNER, A., BEAUCHAMP, J., AND HAKEN, L. Methods for multiple wavetable synthesis of musical instrument tones. *Journal of the Audio Engineering Society 41*, 5 (May 1993), 336–355.

[45] HORNER, A., CHEUNG, N.-M., AND BEAUCHAMP, J. Genetic algorithm optimization of additive synthesis envelope breakpoints and group synthesis parameters. In *Proceedings of the International Computer Music Conference* (1995), International Computer Music Association, pp. 215–222.

[46] HOWE, JR., H. S. Creativity in computer music. *BYTE 4*, 7 (July 1979), 158–173.

[47] JAFFE, D. A. Spectrum analysis tutorial, part 1: The discrete Fourier transform. *Computer Music Journal 11*, 2 (1987), 9–24.

[48] JAFFE, D. A. Spectrum analysis tutorial, part 2: Properties and applications of the discrete Fourier transform. *Computer Music Journal 11*, 3 (1987), 17–35.

[49] JAFFE, D. A. Ten criteria for evaluating synthesis techniques. *Computer Music Journal 19*, 1 (Spring 1995), 76–87.

[50] JENSEN, K. Envelope model of isolated musical sounds. In *Proceedings of the Second COST Workshop on Digital Audio Effects (DAFx99)* (Trondheim, Dec. 9–11 1999), NTNU. Available online at `http://www.tele.ntnu.no/akustikk/meetings/DAFx99/-jensen.pdf` (2002-10-01).

[51] JENSEN, K. The timbre model. In *Proceedings of the Workshop on Current Research Directions in Computer Music* (Barcelona, Nov. 15–17 2001), MOSART. Available online at `http://www.iua.upf.es/mtg/-mosart/papers/p08.pdf` (2002-10-01).

[52] KLECZKOWSKI, P. Group additive synthesis. *Computer Music Journal 13*, 1 (Spring 1989), 12–20.

[53] KODERA, K., GENDRIN, R., AND DE VILLEDARY, C. Analysis of time-varying signals with small *bt* values. *IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-26*, 1 (Feb. 1978), 64–76.

[54] LAUGHLIN, R. G., TRUAX, B. D., AND FUNT, B. V. Synthesis of acoustic timbres using principal component analysis. In *Proceedings of the International Computer Music Conference* (Glasgow, 1990), International Computer Music Association, pp. 95–99.

[55] LUCE, D., AND CLARK, JR., M. Physical correlates of brass-instrument tones. *Journal of the Acoustical Society of America 42*, 6 (1967), 1232–1243.

[56] MAHER, R. C. Evaluation of a method for separating digitized duet signals. *Journal of the Audio Engineering Society 38*, 12 (Dec. 1990), 956–979.

[57] MAHER, R. C., AND BEAUCHAMP, J. An investigation of vocal vibrato for synthesis. *Applied Acoustics 30* (1990), 219–245.

[58] MATHEWS, M. V., AND PIERCE, J. R., Eds. *Current Directions in Computer Music Research*. MIT Press, Cambridge, Massachusetts, 1989.

[59] MCADAMS, S., BEAUCHAMP, J. W., AND MENEGUZZI, S. Discrimination of musical instrument sounds resynthesized with simplified spectrotemporal parameters. *Journal of the Acoustical Society of America 105*, 2 (Feb. 1999), 882–897.

[60] MCAULAY, R. J., AND QUATIERI, T. F. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-34*, 4 (Aug. 1986), 744–754.

[61] MCGEE, W. F., AND MERKLEY, P. A real-time logarithmic-frequency phase vocoder. *Computer Music Journal 15*, 1 (Spring 1991), 20–27.

[62] MOHR, J. Optimized multiple wavetable analysis/synthesis: Results in detail. Tech. Rep. TR02-19, University of Alberta, Edmonton, Alberta, Canada, Sept. 2002. Available online at `http://www.cs.ualberta.ca/-cgi-bin/techreport.cgi?action:menu` (2002-10-01).

[63] MOORE, F. R. An introduction to the mathematics of digital signal processing. In Strawn [91], ch. 1, pp. 1–67. Originally printed in *Computer Music Journal* 2(1):38–47 and 2(2):38–60, 1978.

[64] MOORE, F. R. Table lookup noise for sinusoidal digital oscillators. In Roads and Strawn [81], ch. 20, pp. 326–334. Originally printed in *Computer Music Journal* 1(2): 26–29, 1977.

[65] MOORE, F. R. *Elements of Computer Music.* Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[66] MOORER, J. A. Signal processing aspects of computer music: A survey. In Strawn [91], ch. 4, pp. 149–220. A revised and updated version of an article which first appeared in *Proceedings of the IEEE* 65(8):1108–37, 1977 and in *Computer Music Journal* 1(1):4–37, 1977.

[67] PAVLIDIS, T. Waveform segmentation through functional approximation. *IEEE Transactions on Computers C-22*, 7 (1973), 689–697.

[68] PAVLIDIS, T., AND HOROWITZ, S. L. Segmentation of plane curves. *IEEE Transactions on Computers C-23*, 8 (1974), 860–870.

[69] PEYSER, J. *The New Music: The Sense Behind the Sound.* Dell, New York, 1971.

[70] PLOMP, R. The ear as a frequency analyzer. *Journal of the Acoustical Society of America 36*, 9 (Sept. 1964), 1628–1636.

[71] PLOMP, R. *Aspects of Tone Sensation: A Psychophysical Study.* Academic, London, 1976.

[72] POLANSKY, L., AND ERBE, T. Spectral mutation in Soundhack: A brief description. In *Proceedings of the International Computer Music Conference* (1995), International Computer Music Association, pp. 307–310.

[73] PORTNOFF, M. Implementation of the digital phase vocoder using the fast Fourier transform. *IEEE Transactions on Acoustics, Speech and Signal Processing 24*, 3 (1976), 243–248.

[74] PRESS, W. H., ET AL. *Numerical Recipes in C: The Art of Scientific Computing*, second ed. Cambridge University Press, Cambridge, 1992.

[75] RASCH, R. A., AND PLOMP, R. The perception of musical tones. In Deutsch [19], ch. 1, pp. 1–24.

[76] RISSET, J.-C. Computer study of trumpet tones. Tech. rep., Bell Telephone Laboratories, 1966.

[77] RISSET, J.-C., AND MATHEWS, M. V. Analysis of musical-instrument tones. *Physics Today 22*, 2 (1969), 23–30.

[78] RISSET, J.-C., AND WESSEL, D. L. Exploration of timbre by analysis and synthesis. In Deutsch [19], ch. 2, pp. 25–58.

[79] RISSET, J.-C., AND WESSEL, D. L. Exploration of timbre by analysis and synthesis. In *The Psychology of Music*, D. Deutsch, Ed., 2 ed. Academic, San Diego, 1999, ch. 5, pp. 113–169.

[80] ROADS, C. *The Computer Music Tutorial.* MIT Press, Cambridge, Massachusetts, 1996.

[81] ROADS, C., AND STRAWN, J., Eds. *Foundations of Computer Music.* MIT Press, Cambridge, Massachusetts, 1985.

[82] SALDANHA, E. L., AND CORSO, J. F. Timbre cues and the identification of musical instruments. *Journal of the Acoustical Society of America 36*, 11 (Nov. 1964), 2021–2026.

[83] SANDELL, G., AND MARTENS, W. Prototyping and interpolation of multiple musical timbres using principal components-based analysis. In *Proceedings of the 1992 International Computer Music Conference* (1992), A. Strange, Ed., International Computer Music Association, pp. 34–37.

[84] SCHAFER, R. W., AND RABINER, L. R. Digital representations of speech signals. *Proceedings of the IEEE 36*, 4 (1975), 662–677.

[85] SEASHORE, C. E. *Psychology of Music.* McGraw-Hill, 1938.

[86] SERRA, M.-H., RUBINE, D., AND DANNENBERG, R. Analysis and synthesis of tones by spectral interpolation. *Journal of the Audio Engineering Society 38*, 3 (Mar. 1990), 111–128.

[87] SERRA, X., AND SMITH III, J. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal 14*, 4 (Winter 1990), 12–24.

[88] SMITH III, J. O., AND SERRA, X. PARSHL: An analysis/synthesis program for nonharmonic sounds based on a sinusoidal representation. In *Proceedings of the International Computer Music Conference* (1987), International Computer Music Association.

[89] STAPLETON, J. C., AND BASS, S. C. Synthesis of musical tones based on the Karhunen-Loève transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-36*, 3 (Mar. 1988), 305–319.

[90] STRAWN, J. Approximation and syntactic analysis of amplitude and frequency functions for digital sound synthesis. *Computer Music Journal 4*, 3 (Fall 1980), 3–24.

[91] STRAWN, J., Ed. *Digital Audio Signal Processing: An Anthology.* William Kaufmann, Los Altos, California, 1985.

[92] VERMA, T. S., AND MENG, T. H. Y. Extending spectral modeling synthesis with transient modeling synthesis. *Computer Music Journal 24*, 2 (Summer 2000), 47–59.

[93] WEISS, M. A. *Algorithms, Data Structures, and Problem Solving with C++.* Addison-Wesley, Menlo Park, CA, 1996.

[94] WEISS, M. A. *Data Structures and Algorithm Analysis in C++*, second ed. Addison-Wesley, Menlo Park, CA, 1999.

[95] WELLS, T. H. *The Technique of Electronic Music.* Schirmer Books, New York, 1981.

[96] WRIGHT, M., BEAUCHAMP, J., FITZ, K., RODET, X., RÖBEL, A., SERRA, X., AND WAKEFIELD, G. Analysis/synthesis comparison. *Organised Sound 5*, 3 (2000), 173–189.

[97] YUEN, J., AND HORNER, A. Hybrid sampling-wavetable synthesis with genetic algorithms. *Journal of the Audio Engineering Society 45*, 5 (May 1997), 316–329.

[98] ZHENG, H., AND BEAUCHAMP, J. W. Analysis and critical-band-based group wavetable synthesis of piano tones. In *Proceedings of the International Computer Music Conference* (1999), International Computer Music Association, pp. 9–12.

[99] ZWICKER, E. Subdivision of the audible frequency range into critical bands (*frequenzgruppen*). *Journal of the Acoustical Society of America 33*, 2 (Feb. 1961), 248. A letter to the editor.

[100] ZWICKER, E., AND SCHARF, B. A model of loudness summation. *Psychological Review 72*, 1 (1965), 3–26.

[101] ZWICKER, E., AND ZWICKER, U. T. Audio engineering and psychoacoustics: Matching signals to the final receiver, the human auditory system. *Journal of the Audio Engineering Society 39*, 3 (Mar. 1991), 115–126.

Appendix A

Search Results in Detail

GROUP 1		One-level Exhaustive Search							
		Depth 1		Depth 2		Depth 3		Depth 4	
Instr	Pitch	Time	Error	Time	Error	Time	Error	Time	Error
bsn	A♯1	0.11	136.0	4.96	125.0	112	106.4	1646	85.7
	C♯2	0.12	244.8	5.03	190.6	112	152.8	1650	129.1
	E2	0.15	177.3	4.94	140.9	112	112.1	1647	87.7
	G2	0.13	252.6	5.00	121.6	112	88.8	1643	73.9
	A♯2	0.11	143.6	4.93	113.4	112	74.3	1643	55.4
	C♯3	0.11	182.4	5.09	101.1	112	56.2	1643	37.5
clb	C♯2	0.12	257.3	4.93	231.8	112	198.4	1644	170.4
	E2	0.12	159.8	4.92	141.9	112	129.4	1653	121.6
	G2	0.12	228.6	4.92	175.0	112	147.1	1643	129.4
	A♯2	0.12	214.8	4.95	152.6	112	120.2	1643	106.4
	C♯3	0.12	168.0	4.93	138.8	112	104.9	1643	90.4
hrn	E2	0.12	312.6	4.96	187.0	112	108.6	1643	87.1
	G2	0.12	112.0	4.92	81.5	112	56.1	1642	40.3
	A♯2	0.11	255.6	4.93	107.8	112	73.7	1642	53.4
	C♯3	0.11	94.1	4.93	47.2	112	30.0	1643	21.6
pno	A♯1	0.11	133.6	4.92	91.7	112	65.1	1643	51.5
	C♯2	0.11	152.1	4.92	110.5	112	98.5	1643	90.0
	E2	0.11	291.0	4.93	204.4	112	155.7	1642	112.9
	G2	0.12	273.7	4.94	166.2	112	95.4	1644	75.4
	A♯2	0.12	229.7	4.96	162.2	112	134.1	1643	105.8
	C♯3	0.12	126.7	4.93	60.1	112	35.7	1644	27.5
sax	A♯1	0.11	107.5	4.93	85.5	112	71.6	1639	58.4
	C♯2	0.12	135.9	4.92	119.6	112	97.2	1634	78.4
	E2	0.12	157.3	4.97	138.8	112	122.2	1636	108.0
	G2	0.12	138.1	4.96	109.9	112	82.7	1636	65.7
	A♯2	0.11	113.3	4.97	89.7	112	72.0	1635	58.5
	C♯3	0.11	178.6	4.95	131.4	112	104.1	1658	84.0
trb	E2	0.12	147.4	4.93	96.1	112	71.6	1671	58.4
	G2	0.11	150.5	4.94	104.5	112	72.5	1633	55.7
	A♯2	0.11	126.6	4.93	80.0	112	54.6	1633	40.3
	C♯3	0.11	296.4	4.93	164.8	112	124.5	1640	79.5
vla	C♯3	0.27	366.1	11.11	297.5	251	239.6	3642	200.9
vlb	A♯1	0.23	181.8	9.71	124.0	220	86.2	3184	63.7
	C♯2	0.28	186.8	11.11	119.2	252	85.6	3634	72.4
	E2	0.23	169.6	9.73	128.9	220	106.9	3183	90.2
	G2	0.23	100.6	9.74	70.6	221	54.7	3184	46.0
	A♯2	0.25	325.6	10.52	165.0	240	113.9	3443	85.6
	C♯3	0.23	365.8	9.80	252.0	219	131.5	3183	97.7
vlc	C♯2	0.24	171.8	10.42	118.3	231	87.4	3376	71.2
	E2	0.23	110.5	9.73	74.7	217	60.7	3182	52.6
	G2	0.23	204.4	9.74	103.2	217	74.8	3181	55.3
	A♯2	0.23	362.3	9.73	174.0	217	104.2	3182	74.3
	C♯3	0.25	200.9	10.98	116.3	244	78.2	3584	62.7
Mean		0.15	196.4	6.41	132.9	145	98.6	2114	79.4
Std. Dev.		0.06	76.3	2.40	50.9	54	40.4	772	35.2

Table A.1: Group 1, one-level exhaustive search.

GROUP 1 Depth + 1		Augmented Search								
		2+1			3+1			4+1		
Instr	Pitch	T_{aug}	T_{srch}	Error	T_{aug}	T_{srch}	Error	T_{aug}	T_{srch}	Error
bsn	A♯1	4.10	9.1	108.4	6.6	119	89.0	8.6	1655	75.1
	C♯2	2.86	7.9	153.5	6.9	119	132.6	9.4	1659	110.3
	E2	2.34	7.3	117.7	5.0	117	91.2	8.4	1655	77.8
	G2	4.38	9.3	92.0	8.1	121	75.6	10.8	1654	64.5
	A♯2	3.51	8.5	75.9	7.7	120	59.4	10.3	1653	46.7
	C♯3	2.93	7.9	56.8	7.3	119	38.6	11.3	1654	31.2
clb	C♯2	2.05	7.0	207.4	3.9	116	171.2	4.7	1649	160.0
	E2	4.09	9.0	133.1	7.3	120	123.9	9.9	1663	117.3
	G2	3.21	8.2	148.1	5.4	118	134.0	8.5	1651	121.5
	A♯2	3.51	8.5	120.6	5.8	118	109.9	9.9	1653	98.5
	C♯3	2.64	7.6	107.7	6.5	119	93.4	8.9	1652	82.3
hrn	E2	2.93	7.9	109.2	4.6	117	92.3	8.0	1651	72.5
	G2	2.67	7.6	60.4	6.6	119	43.2	10.3	1652	32.7
	A♯2	2.63	7.6	74.0	5.0	117	55.4	7.5	1650	40.9
	C♯3	3.80	8.8	30.1	7.0	119	22.8	10.3	1653	16.7
pno	A♯1	2.93	7.9	67.6	6.2	118	53.5	9.4	1652	46.5
	C♯2	4.09	9.0	99.5	8.9	121	92.0	10.8	1654	82.8
	E2	3.51	8.5	159.6	5.8	118	124.2	7.5	1650	91.3
	G2	3.55	8.5	108.6	6.2	118	77.1	8.9	1653	64.9
	A♯2	3.52	8.5	139.2	6.6	119	108.9	9.9	1653	82.7
	C♯3	4.39	9.3	38.8	8.5	121	27.7	9.4	1653	23.1
sax	A♯1	4.39	9.3	72.7	7.0	119	59.5	10.3	1649	52.6
	C♯2	4.10	9.1	101.5	6.6	119	83.1	10.3	1644	65.7
	E2	3.51	8.6	122.6	5.4	118	111.4	6.6	1643	99.7
	G2	3.81	8.8	83.7	7.0	119	70.8	9.4	1645	57.2
	A♯2	4.44	9.5	75.0	7.3	119	63.2	11.3	1646	50.6
	C♯3	4.79	9.8	105.4	6.6	119	86.0	10.8	1669	72.3
trb	E2	2.64	7.6	71.6	5.8	118	58.8	8.9	1680	49.1
	G2	3.54	8.5	72.5	6.6	119	56.0	8.4	1641	45.7
	A♯2	4.38	9.3	54.7	7.3	120	42.2	10.8	1644	30.4
	C♯3	2.34	7.3	125.8	5.8	118	79.8	5.6	1646	65.1
vla	C♯3	14.37	25.5	252.6	31.7	283	207.9	52.8	3695	180.6
vlb	A♯1	13.14	22.9	87.5	29.2	249	66.0	40.9	3225	54.5
	C♯2	15.16	26.3	86.1	42.0	294	73.7	53.9	3688	65.8
	E2	14.93	24.8	107.8	32.3	252	93.7	36.3	3219	82.0
	G2	17.93	27.8	54.9	32.4	254	47.3	41.8	3226	41.6
	A♯2	11.08	21.7	114.1	23.6	263	87.4	35.3	3478	68.1
	C♯3	5.18	15.1	131.5	14.0	233	100.7	24.7	3208	80.8
vlc	C♯2	9.21	19.8	88.4	21.9	253	73.4	39.1	3415	63.0
	E2	11.53	21.4	60.8	31.0	248	53.6	41.4	3223	49.2
	G2	14.52	24.4	75.0	28.8	246	56.9	42.0	3223	46.3
	A♯2	5.18	15.0	107.0	8.1	225	78.6	26.4	3208	62.9
	C♯3	10.33	21.4	79.1	27.4	272	63.2	49.2	3633	53.0
Mean		5.82	12.3	100.9	12.2	157	82.1	17.9	2132	69.2
Std. Dev.		4.36	6.6	42.1	10.4	63	36.2	14.9	786	33.0

Table A.2: Group 1, two-level exhaustive search with 1-table augmentation.

GROUP 1 Depth + 2		Augmented Search								
		1+2			2+2			3+2		
Instr	Pitch	T_{aug}	T_{srch}	Error	T_{aug}	T_{srch}	Error	T_{aug}	T_{srch}	Error
bsn	A♯1	4.80	4.92	108.4	125	130	89.2	180	292	70.3
	C♯2	3.61	3.73	153.5	90	95	131.2	191	303	112.0
	E2	2.81	2.92	117.7	72	77	88.0	137	249	75.2
	G2	4.89	5.00	92.0	134	139	73.9	222	335	63.2
	A♯2	4.41	4.53	75.9	108	112	55.6	211	323	45.9
	C♯3	3.60	3.72	56.8	89	94	37.6	201	313	29.1
clb	C♯2	2.20	2.32	208.8	62	67	188.0	107	219	160.3
	E2	4.29	4.40	132.5	124	129	122.2	201	313	115.9
	G2	3.53	3.64	152.0	99	104	131.0	148	260	120.5
	A♯2	4.12	4.24	120.6	107	112	109.4	158	270	97.4
	C♯3	2.80	2.92	110.6	82	87	93.3	179	291	82.2
hrn	E2	4.02	4.13	109.2	89	94	89.2	126	238	73.0
	G2	3.32	3.43	60.4	80	85	41.8	179	291	32.5
	A♯2	3.93	4.05	73.9	80	85	53.8	137	249	39.5
	C♯3	4.71	4.83	30.1	115	120	21.6	191	303	17.4
pno	A♯1	3.60	3.72	67.6	89	94	52.3	170	282	46.1
	C♯2	4.71	4.83	99.8	124	129	91.2	244	356	84.0
	E2	5.36	5.48	163.4	106	111	122.7	159	271	101.4
	G2	4.53	4.65	111.6	107	112	81.3	169	281	64.0
	A♯2	5.34	5.46	143.0	108	113	113.1	180	292	83.6
	C♯3	5.30	5.42	38.8	136	141	27.5	235	347	22.7
sax	A♯1	5.31	5.42	73.0	132	137	59.7	191	303	52.5
	C♯2	4.79	4.91	101.5	124	129	82.2	180	292	68.0
	E2	4.29	4.40	122.5	106	111	108.7	148	260	95.0
	G2	4.29	4.40	83.9	115	120	68.4	190	302	58.9
	A♯2	5.51	5.62	75.0	133	138	62.2	200	312	49.2
	C♯3	6.01	6.13	105.2	141	146	84.1	179	291	70.9
trb	E2	3.31	3.43	71.6	80	84	58.4	158	270	49.3
	G2	4.61	4.73	72.5	106	111	55.8	180	292	43.5
	A♯2	5.72	5.84	54.7	133	138	40.4	202	314	30.9
	C♯3	3.13	3.25	125.8	71	76	80.2	158	270	61.7
vla	C♯3	17.04	17.31	252.2	432	443	204.5	865	1116	179.3
vlb	A♯1	15.88	16.11	87.5	395	405	64.2	793	1013	52.9
	C♯2	17.49	17.75	89.0	452	463	72.7	1137	1389	63.8
	E2	18.88	19.10	107.8	447	456	90.6	878	1098	82.7
	G2	20.53	20.76	54.9	533	543	46.2	859	1080	40.6
	A♯2	12.96	13.21	114.1	335	346	85.7	637	877	67.6
	C♯3	6.24	6.52	131.5	155	165	97.8	388	607	81.7
vlc	C♯2	10.51	10.75	88.4	275	285	72.8	603	834	62.6
	E2	13.08	13.31	60.8	344	353	52.6	856	1073	48.2
	G2	16.33	16.56	75.0	430	439	55.7	792	1009	45.1
	A♯2	7.88	8.11	115.6	156	165	78.1	223	440	59.7
	C♯3	13.19	13.45	80.5	309	320	63.2	752	996	50.7
Mean		7.04	7.20	101.6	175	181	81.4	332	477	68.6
Std. Dev.		5.07	5.12	42.5	130	132	37.1	282	331	33.2

Table A.3: Group 1, two-level exhaustive search with 2-table augmentation.

GROUP 1		Genetic Algorithm Search (Thorough)								
		2+0			3+0			4+0		
Instr	Pitch	T_{GA}	Gen	Error	T_{GA}	Gen	Error	T_{GA}	Gen	Error
bsn	A♯1	39.3	576	267.7	76.8	648	156.2	149.1	806	106.0
	C♯2	51.9	757	239.9	110.8	965	171.1	163.5	967	158.2
	E2	29.7	395	480.6	78.0	653	133.6	168.6	981	112.3
	G2	31.0	425	269.8	84.2	725	120.1	137.7	741	209.4
	A♯2	52.2	772	120.0	109.4	973	95.4	185.5	1058	81.9
	C♯3	35.9	525	387.7	87.9	747	134.2	191.9	1141	61.8
clb	C♯2	35.0	476	258.3	94.9	815	228.9	119.5	631	217.9
	E2	32.4	443	168.3	72.7	603	162.5	94.4	434	163.6
	G2	50.0	736	192.9	97.9	855	171.2	145.6	805	160.1
	A♯2	51.0	738	169.4	81.9	690	152.4	132.4	718	125.1
	C♯3	42.3	609	158.3	107.6	931	126.2	113.4	554	138.0
hrn	E2	44.0	620	233.7	122.6	1112	150.0	163.0	947	122.4
	G2	63.8	931	86.5	92.7	809	76.2	199.1	1139	53.6
	A♯2	53.0	833	133.4	107.8	984	102.4	155.5	915	79.0
	C♯3	49.1	685	83.5	77.4	646	83.9	179.9	1065	30.4
pno	A♯1	60.1	876	102.7	92.0	800	91.5	136.7	736	71.8
	C♯2	28.2	365	175.2	80.4	675	107.3	102.5	527	103.9
	E2	33.7	452	238.8	62.9	496	201.3	114.8	549	170.1
	G2	16.6	197	216.9	73.1	612	120.5	151.2	820	99.7
	A♯2	42.5	599	196.1	86.3	733	158.6	119.2	599	134.7
	C♯3	62.3	857	72.8	60.8	468	60.9	206.0	1231	33.4
sax	A♯1	42.1	570	155.9	104.0	899	81.7	116.3	584	96.6
	C♯2	48.2	703	130.7	66.3	536	145.1	143.6	776	106.0
	E2	32.5	471	195.9	69.3	557	141.9	111.3	546	129.7
	G2	44.0	641	122.0	93.6	800	98.6	143.5	777	91.4
	A♯2	9.2	102	475.2	93.9	793	108.1	178.8	1023	68.5
	C♯3	46.4	648	157.3	56.4	430	213.7	134.6	722	117.2
trb	E2	48.4	710	109.9	109.4	970	89.2	161.3	907	86.1
	G2	47.7	711	164.8	122.3	1117	89.4	212.2	1291	78.5
	A♯2	40.3	586	165.2	69.4	565	112.6	174.6	972	60.4
	C♯3	32.8	487	203.3	99.0	879	148.2	128.6	711	128.1
vla	C♯3	234.6	1347	349.1	288.3	978	373.0	339.4	782	328.3
vlb	A♯1	180.6	1247	179.8	310.6	1268	157.1	380.6	1058	161.5
	C♯2	195.6	1113	269.7	412.3	1444	183.5	768.9	1969	103.9
	E2	87.8	595	171.3	195.8	764	145.5	360.9	988	125.7
	G2	197.1	1389	90.5	325.6	1320	76.9	395.9	1112	68.7
	A♯2	275.1	1877	239.5	361.9	1320	221.7	632.1	1707	159.5
	C♯3	178.2	1265	310.7	334.2	1375	251.5	575.6	1675	156.8
vlc	C♯2	176.1	1091	154.2	333.2	1268	123.0	400.1	1029	121.4
	E2	129.7	871	379.2	356.5	1487	78.7	388.1	1077	146.6
	G2	171.2	1178	154.6	341.5	1411	108.1	499.3	1421	93.4
	A♯2	137.5	980	238.7	209.1	827	193.8	461.0	1301	132.9
	C♯3	165.1	964	178.8	408.4	1477	126.7	650.2	1655	99.2
Mean		79.6	754	205.8	153.9	894	141.2	243.9	964	118.5
Std. Dev.		67.1	342	96.4	112.9	299	58.0	171.1	346	53.2

Table A.4: Genetic algorithm search on Group 1, with population size of 100 and termination on convergence after 50 generations (thorough version).

GROUP 1		Augmented Genetic Algorithm Search (Thorough)								
		2+1			3+1			4+1		
Instr	Pitch	T_{aug}	T_{srch}	Error	T_{aug}	T_{srch}	Error	T_{aug}	T_{srch}	Error
bsn	A♯1	6.0	45.3	107.5	17.4	94.2	89.5	21.3	170.4	79.8
	C♯2	3.9	55.8	153.7	9.3	120.1	133.6	11.3	174.8	117.3
	E2	12.8	42.4	112.1	17.4	95.5	99.4	10.8	179.4	84.3
	G2	5.3	36.3	93.8	34.4	118.5	76.4	31.4	169.1	72.1
	A♯2	5.1	57.3	75.1	8.9	118.2	63.8	11.3	196.8	54.0
	C♯3	5.4	41.3	57.1	9.5	97.4	43.7	11.3	203.2	34.5
clb	C♯2	6.3	41.2	200.1	8.9	103.8	189.0	11.3	130.8	183.9
	E2	6.3	38.7	132.3	9.3	81.9	124.7	11.4	105.8	120.4
	G2	12.4	62.4	148.5	8.5	106.4	136.2	21.5	167.1	126.3
	A♯2	5.1	56.1	120.5	9.3	91.2	112.3	21.6	154.0	102.5
	C♯3	12.7	55.0	108.2	9.3	116.9	93.7	11.4	124.8	90.5
hrn	E2	5.1	49.1	109.7	7.7	130.4	90.8	11.3	174.3	80.0
	G2	4.2	68.0	60.4	8.9	101.6	48.0	11.2	210.3	38.1
	A♯2	4.2	57.2	73.7	8.5	116.3	54.8	11.4	166.9	42.6
	C♯3	6.3	55.3	30.1	9.3	86.7	24.3	11.2	191.1	18.5
pno	A♯1	6.1	66.2	67.8	9.3	101.3	55.6	11.3	148.0	49.7
	C♯2	7.2	35.5	102.3	9.3	89.7	94.1	11.4	113.9	89.1
	E2	6.2	39.9	167.8	9.3	72.2	145.8	11.2	126.0	116.4
	G2	7.1	23.7	104.1	8.9	82.0	79.3	21.5	172.7	66.3
	A♯2	6.0	48.5	134.8	9.3	95.5	115.7	11.4	130.6	94.8
	C♯3	6.4	68.6	36.2	9.3	70.1	31.5	21.4	227.4	25.4
sax	A♯1	8.0	50.1	73.9	9.3	113.3	63.5	22.1	138.4	58.5
	C♯2	6.0	54.2	100.1	9.3	75.6	85.2	21.4	165.0	80.4
	E2	5.4	37.8	123.0	9.3	78.6	115.8	21.9	133.2	110.5
	G2	5.4	49.4	84.7	8.9	102.5	73.2	11.6	155.1	64.8
	A♯2	6.9	16.1	76.4	9.3	103.2	67.2	11.3	190.1	54.0
	C♯3	5.4	51.8	108.2	9.3	65.7	98.8	31.5	166.1	84.1
trb	E2	4.8	53.2	72.3	17.4	126.8	59.9	21.2	182.5	52.6
	G2	6.0	53.8	73.6	8.1	130.4	58.7	11.2	223.4	51.9
	A♯2	5.4	45.7	61.5	9.3	78.7	48.6	11.3	185.9	36.4
	C♯3	4.8	37.6	126.0	8.5	107.5	88.5	21.4	150.0	67.1
vla	C♯3	23.1	257.7	243.7	64.5	352.8	215.9	57.1	396.5	201.2
vlb	A♯1	19.9	200.5	88.5	36.1	346.7	70.7	43.5	424.1	60.5
	C♯2	29.1	224.7	86.1	47.2	459.6	75.5	79.2	848.1	70.2
	E2	23.9	111.7	109.9	36.1	231.9	97.6	43.2	404.1	89.9
	G2	18.6	215.8	56.7	36.3	361.9	49.6	62.6	458.5	45.1
	A♯2	17.8	292.9	114.6	42.3	404.2	90.4	94.0	726.1	73.8
	C♯3	17.3	195.5	131.5	33.9	368.1	106.6	43.5	619.1	88.1
vlc	C♯2	27.1	203.1	88.6	40.0	373.2	74.7	49.0	449.1	68.2
	E2	34.1	163.8	60.7	33.9	390.4	54.9	61.7	449.8	52.1
	G2	22.2	193.4	77.7	34.6	376.1	60.5	43.2	542.5	54.7
	A♯2	31.2	168.7	111.6	49.7	258.8	79.3	43.5	504.5	64.6
	C♯3	32.1	197.2	80.2	81.9	490.3	64.6	54.9	705.1	58.3
Mean		11.5	91.1	101.1	20.2	174.1	86.1	27.2	271.0	76.1
Std. Dev.		9.0	74.3	40.9	17.7	128.3	38.2	20.8	189.5	36.6

Table A.5: Augmented genetic algorithm search on Group 1, thorough version.

GROUP 1		Genetic Algorithm Search (Quick)								
		2+0			3+0			4+0		
Instr	Pitch	T_{GA}	Gen	Error	T_{GA}	Gen	Error	T_{GA}	Gen	Error
bsn	A♯1	5.8	147	598.1	16.1	247	522.4	26.8	256	393.1
	C♯2	14.2	403	331.5	23.2	383	389.8	18.7	153	523.0
	E2	6.0	147	997.4	15.5	230	735.9	24.4	204	763.1
	G2	5.2	125	1246.0	23.2	341	536.9	33.1	332	577.5
	A♯2	3.1	63	1402.3	26.9	462	605.2	29.4	319	555.3
	C♯3	2.9	58	1241.2	15.2	219	495.9	33.7	388	248.5
clb	C♯2	13.5	370	326.7	22.3	345	270.9	26.3	255	262.4
	E2	6.8	177	394.9	25.5	420	236.0	32.8	320	207.3
	G2	7.9	209	450.8	20.8	335	229.1	36.2	395	197.2
	A♯2	9.0	230	446.8	15.7	231	270.6	37.0	390	208.2
	C♯3	8.7	235	380.7	18.4	289	287.3	32.1	342	173.2
hrn	E2	13.5	382	321.3	28.0	493	212.6	25.7	225	262.1
	G2	12.6	361	217.7	26.9	487	173.2	32.0	320	168.6
	A♯2	14.3	412	191.5	25.5	433	136.8	34.5	359	139.9
	C♯3	17.2	498	133.2	22.6	371	196.6	27.4	244	182.0
pno	A♯1	7.3	181	229.9	7.4	63	226.7	14.5	72	179.6
	C♯2	9.9	257	175.8	23.1	376	142.1	32.0	314	118.0
	E2	3.3	70	319.2	6.1	33	809.2	25.8	247	193.3
	G2	4.8	98	249.4	6.2	41	238.9	25.5	217	147.2
	A♯2	2.8	51	385.7	14.1	202	202.5	29.0	272	166.0
	C♯3	12.3	334	129.1	21.2	342	98.2	28.6	272	75.3
sax	A♯1	7.3	189	357.6	19.6	317	174.8	28.2	249	242.0
	C♯2	10.3	288	319.7	32.7	588	170.8	18.1	103	297.1
	E2	5.6	137	587.6	16.3	233	349.6	26.9	250	268.6
	G2	12.7	340	307.2	14.5	201	310.2	36.7	420	182.3
	A♯2	8.0	207	396.3	24.1	402	264.9	34.9	370	161.5
	C♯3	14.1	389	280.5	5.7	24	675.4	16.0	79	393.4
trb	E2	9.8	275	358.8	17.9	263	326.0	43.9	521	202.5
	G2	14.3	380	257.8	12.1	138	442.6	27.0	261	282.3
	A♯2	8.4	219	340.2	28.4	479	185.7	22.1	178	319.5
	C♯3	6.7	164	399.0	26.1	434	199.6	27.1	264	213.3
vla	C♯3	33.1	364	787.3	52.3	335	728.1	116.1	561	484.9
vlb	A♯1	24.4	318	720.5	42.2	314	564.1	35.8	137	582.1
	C♯2	48.6	565	829.4	68.2	463	578.6	99.7	452	504.9
	E2	17.6	217	237.6	35.1	247	195.5	36.9	110	197.8
	G2	45.9	625	140.5	65.1	541	139.8	66.7	337	115.2
	A♯2	37.0	454	1033.1	70.1	523	674.1	103.4	519	527.4
	C♯3	21.7	280	800.2	48.0	394	511.9	86.4	461	415.3
vlc	C♯2	24.4	299	590.6	75.4	582	309.6	34.0	69	501.4
	E2	26.8	331	866.0	52.0	419	573.6	67.1	345	429.7
	G2	26.2	323	292.2	53.3	429	207.2	65.1	316	172.0
	A♯2	28.6	357	322.7	68.2	545	244.9	74.8	387	187.4
	C♯3	22.0	227	326.3	102.0	720	166.6	31.5	56	335.4
Mean		14.8	273	481.9	31.0	347	337.5	39.6	287	296.7
Std. Dev.		11.2	134	58.0	21.8	157	183.2	24.0	126	161.0

Table A.6: Genetic algorithm search on Group 1, with population size of 50 and termination on convergence after 25 generations (quick version).

GROUP 1		Augmented Genetic Algorithm Search (Quick)								
		2+1			3+1			4+1		
Instr	Pitch	T_{aug}	T_{srch}	Error	T_{aug}	T_{srch}	Error	T_{aug}	T_{srch}	Error
bsn	A♯1	13.4	19.2	106.4	34.6	50.7	88.0	11.3	38.1	84.9
	C♯2	5.3	19.5	154.0	9.4	32.6	140.7	172.7	191.4	122.8
	E2	13.7	19.8	112.1	18.0	33.5	100.2	21.3	45.7	88.4
	G2	13.8	19.0	94.7	17.9	41.0	81.9	21.4	54.5	72.7
	A♯2	6.8	10.0	88.7	9.2	36.0	72.5	11.3	40.7	57.2
	C♯3	6.5	9.4	72.9	9.0	24.2	42.2	11.2	44.9	37.3
clb	C♯2	6.5	20.0	209.1	17.4	39.6	194.1	31.8	58.1	169.5
	E2	6.8	13.6	132.8	9.6	35.1	126.1	41.4	74.2	119.0
	G2	6.5	14.4	155.1	9.7	30.5	140.4	11.2	47.4	133.3
	A♯2	6.8	15.8	127.0	9.2	24.9	118.5	11.2	48.2	105.3
	C♯3	6.8	15.4	108.9	18.1	36.6	102.8	183.9	216.0	87.4
hrn	E2	6.2	19.7	109.5	8.8	36.8	95.8	21.2	46.9	94.6
	G2	6.5	19.1	61.0	9.2	36.2	48.0	11.2	43.2	41.3
	A♯2	6.2	20.5	77.6	9.6	35.0	58.2	41.5	76.0	52.5
	C♯3	6.1	23.3	30.1	9.2	31.7	25.5	11.2	38.6	23.5
pno	A♯1	6.8	14.1	69.1	17.8	25.3	58.3	21.6	36.1	58.1
	C♯2	6.9	16.8	100.3	9.4	32.5	95.0	31.5	63.5	88.6
	E2	14.3	17.6	173.0	35.2	41.3	150.5	11.2	37.0	120.8
	G2	7.3	12.0	105.8	9.6	15.8	95.8	11.2	36.7	80.2
	A♯2	13.7	16.5	137.2	9.2	23.3	124.2	11.2	40.2	110.5
	C♯3	6.8	19.1	39.8	9.1	30.4	33.4	21.2	49.8	28.1
sax	A♯1	7.1	14.3	74.8	18.0	37.6	64.2	11.2	39.4	62.7
	C♯2	6.2	16.5	101.5	9.4	42.1	90.7	41.3	59.4	84.1
	E2	7.0	12.6	125.5	9.2	25.5	120.4	30.8	57.7	107.5
	G2	6.5	19.2	86.4	9.7	24.2	78.3	11.2	47.9	68.1
	A♯2	7.6	15.6	77.4	9.1	33.1	70.8	11.2	46.1	58.2
	C♯3	6.8	20.8	110.1	17.8	23.5	107.0	181.9	197.9	89.6
trb	E2	6.0	15.9	78.1	9.2	27.1	72.4	30.8	74.7	56.0
	G2	6.5	20.9	75.2	9.4	21.4	70.6	11.2	38.2	61.7
	A♯2	13.2	21.6	55.0	17.6	46.0	49.2	31.3	53.4	45.8
	C♯3	7.2	14.0	132.0	9.2	35.2	86.1	11.2	38.3	72.9
vla	C♯3	34.3	67.5	250.6	65.1	117.4	219.3	122.9	239.0	198.5
vlb	A♯1	25.9	50.4	91.3	52.1	94.3	75.3	101.4	137.2	68.3
	C♯2	30.3	78.9	88.1	46.6	114.9	78.5	100.7	200.4	74.6
	E2	27.3	45.0	112.8	52.4	87.5	100.2	120.7	157.6	91.0
	G2	24.1	70.1	55.8	52.4	117.6	51.5	62.6	129.3	45.5
	A♯2	28.7	65.7	115.5	40.9	111.0	105.1	134.3	237.7	90.9
	C♯3	26.0	47.7	138.2	50.6	98.6	119.8	62.6	149.0	96.7
vlc	C♯2	27.7	52.2	92.2	57.5	132.8	77.0	110.5	144.5	73.3
	E2	25.7	52.5	62.9	36.2	88.2	57.0	392.3	459.4	50.3
	G2	26.1	52.3	80.6	56.3	109.6	68.9	62.6	127.7	55.4
	A♯2	23.8	52.3	108.0	53.4	121.6	89.8	101.4	176.2	66.2
	C♯3	46.6	68.6	87.8	45.0	147.0	69.3	163.0	194.5	62.9
Mean		13.8	28.6	103.8	23.6	54.6	91.0	61.1	100.8	80.4
Std. Dev.		10.3	19.9	41.8	18.4	37.6	38.9	74.5	86.0	34.8

Table A.7: Augmented genetic algorithm search on Group 1, quick version.

Appendix B

Optimization Results in Detail

GROUP 1 3 Oscillators		Three-Table Match Optimization								
		3+0			2+1			1+2		
Instr	Pitch	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
bsn	A♯1	2.90	115	114.5	2.61	11.7	116.6	2.49	7.39	116.8
	C♯2	1.74	114	171.7	1.80	9.7	171.7	1.79	5.49	171.7
	E2	1.31	113	118.3	1.50	8.8	123.1	1.37	4.27	123.1
	G2	2.25	115	103.6	2.32	11.6	106.8	2.32	7.32	106.8
	A♯2	2.50	114	90.1	2.60	11.1	90.4	2.59	7.09	90.4
	C♯3	1.96	114	69.7	1.69	9.6	69.7	1.68	5.38	69.7
clb	C♯2	1.31	113	206.9	0.49	7.5	214.2	0.50	2.80	214.2
	E2	3.25	115	136.6	2.31	11.3	136.2	1.68	6.08	134.9
	G2	1.95	114	161.3	1.57	9.8	160.2	0.57	4.17	158.1
	A♯2	1.52	114	135.3	1.23	9.7	135.3	0.95	5.15	135.3
	C♯3	1.77	114	121.9	1.29	8.9	118.4	1.44	4.34	119.7
hrn	E2	1.19	113	121.4	1.08	9.0	121.3	1.05	5.15	121.2
	G2	2.02	114	65.7	1.55	9.2	65.6	1.61	5.01	65.6
	A♯2	0.89	113	86.6	0.89	8.5	87.8	0.96	4.96	86.6
	C♯3	2.52	115	39.1	2.17	11.0	39.1	2.29	7.09	39.1
pno	A♯1	1.49	113	74.2	1.20	9.1	75.0	1.18	4.88	75.0
	C♯2	4.03	116	109.0	2.81	11.8	107.8	2.20	7.00	107.9
	E2	1.99	114	188.3	1.06	9.6	188.3	1.29	6.79	181.3
	G2	2.75	115	115.6	1.81	10.3	137.5	1.26	5.96	128.1
	A♯2	1.68	114	155.8	1.39	9.9	152.5	1.88	7.38	152.5
	C♯3	2.22	114	49.3	2.01	11.3	51.2	2.02	7.42	51.2
sax	A♯1	2.32	114	82.4	1.76	11.1	83.2	1.76	7.16	83.2
	C♯2	2.39	114	108.1	1.54	10.6	107.5	1.41	6.31	107.5
	E2	1.91	114	129.0	1.39	10.0	129.0	1.28	5.68	129.0
	G2	2.30	114	95.4	2.16	11.0	95.4	1.72	6.12	95.4
	A♯2	1.77	114	84.2	1.53	11.0	82.3	1.54	7.14	82.3
	C♯3	2.06	114	120.7	2.11	11.9	118.4	2.06	8.16	118.2
trb	E2	1.04	113	79.2	0.96	8.6	79.2	0.91	4.31	79.2
	G2	1.96	114	83.7	1.60	10.1	83.7	1.60	6.30	83.7
	A♯2	1.64	114	65.0	1.59	10.9	65.0	1.58	7.38	65.0
	C♯3	2.21	114	141.7	1.33	8.6	141.4	1.40	4.60	138.0
vla	C♯3	4.84	256	270.6	4.75	30.2	269.8	4.94	22.24	268.8
vlb	A♯1	4.85	225	107.7	4.36	27.3	108.7	4.35	20.45	108.7
	C♯2	7.49	259	103.1	6.72	33.0	103.0	6.87	24.67	103.1
	E2	6.68	227	123.5	4.24	29.0	121.7	4.26	23.36	121.7
	G2	6.58	228	69.5	6.44	34.2	69.5	6.07	26.87	69.5
	A♯2	2.83	243	127.2	2.78	24.5	127.2	2.85	16.05	127.2
	C♯3	2.29	221	162.6	1.86	17.0	162.6	1.88	8.38	162.6
vlc	C♯2	4.70	236	105.0	4.66	24.5	105.7	4.11	14.91	105.7
	E2	7.01	224	71.2	6.87	28.3	71.2	6.93	20.23	71.2
	G2	5.14	222	96.4	5.33	29.7	96.2	5.11	21.71	96.2
	A♯2	2.06	219	122.6	1.68	16.7	117.3	2.07	10.17	126.5
	C♯3	5.93	250	95.9	5.64	27.0	95.9	5.08	18.48	96.4
	Mean	2.87	148	113.5	2.48	14.8	114.0	2.39	9.58	113.7
	Std. Dev.	1.75	55	43.5	1.68	8.1	43.7	1.66	6.67	43.2

Table B.1: Group 1, 3-oscillator, 3-table match optimization.

GROUP 1 3 Oscillators		One- and Two-Table Match Optimization								
		2+0			1+1			1+0		
Instr	Pitch	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
bsn	A♯1	0.49	5.45	124.0	0.49	1.23	124.2	0.70	0.81	124.1
	C♯2	0.34	5.37	180.2	0.34	1.08	180.2	0.77	0.89	180.2
	E2	0.24	5.18	137.6	0.21	0.74	137.6	0.77	0.92	137.6
	G2	0.44	5.44	117.5	0.39	0.92	117.5	0.79	0.92	117.5
	A♯2	1.55	6.48	94.9	1.55	2.50	94.9	2.30	2.41	94.9
	C♯3	1.46	6.55	72.1	1.45	2.19	72.1	1.77	1.88	72.1
clb	C♯2	0.74	5.67	215.0	0.56	1.09	214.2	0.98	1.10	214.2
	E2	1.18	6.10	137.3	1.04	1.58	137.2	1.45	1.57	137.2
	G2	0.96	5.88	160.9	0.64	1.59	161.0	1.46	1.58	161.0
	A♯2	0.89	5.84	135.3	1.26	2.21	135.3	1.75	1.87	135.3
	C♯3	1.00	5.93	133.4	0.90	1.43	133.4	1.35	1.47	133.4
hrn	E2	1.15	6.11	121.6	1.13	2.28	121.5	1.23	1.35	162.3
	G2	0.90	5.82	70.4	0.90	1.63	70.4	1.01	1.13	70.4
	A♯2	1.38	6.31	82.9	1.23	2.59	82.9	0.85	0.96	88.1
	C♯3	2.45	7.38	39.3	1.91	2.86	39.3	2.37	2.48	39.3
pno	A♯1	0.25	5.17	81.0	0.21	0.95	81.0	0.69	0.80	81.0
	C♯2	0.55	5.47	108.2	0.41	1.36	108.3	0.88	0.99	108.3
	E2	1.68	6.61	191.0	1.17	3.37	189.7	1.80	1.91	200.2
	G2	0.28	5.22	148.8	0.22	1.58	150.3	1.08	1.20	152.6
	A♯2	1.45	6.41	156.7	1.48	3.05	172.5	1.70	1.82	172.8
	C♯3	0.48	5.41	53.5	0.49	1.43	53.5	1.34	1.46	53.5
sax	A♯1	0.46	5.39	83.2	0.45	1.40	85.7	0.59	0.70	86.3
	C♯2	0.32	5.24	115.5	0.33	1.07	115.5	0.50	0.62	115.5
	E2	0.22	5.19	136.5	0.22	0.76	136.6	0.64	0.76	136.6
	G2	1.24	6.20	97.9	1.18	1.72	97.9	1.63	1.75	97.8
	A♯2	2.19	7.16	84.3	2.38	3.53	84.3	2.21	2.32	84.4
	C♯3	0.65	5.60	124.9	0.31	1.75	126.9	0.90	1.01	126.9
trb	E2	0.21	5.14	89.1	0.25	1.16	84.9	0.69	0.81	84.9
	G2	0.26	5.20	90.8	0.26	1.42	90.8	0.70	0.81	96.9
	A♯2	0.38	5.31	73.3	0.39	1.85	73.3	0.60	0.71	82.4
	C♯3	0.87	5.80	137.6	0.84	2.00	137.6	1.63	1.74	137.6
vla	C♯3	0.87	11.98	286.8	0.77	3.40	286.7	2.16	2.43	287.0
vlb	A♯1	0.73	10.44	117.4	0.74	3.44	117.4	1.22	1.45	118.5
	C♯2	1.02	12.13	106.5	1.10	4.19	116.5	2.35	2.63	116.4
	E2	1.05	10.78	125.8	1.01	4.91	125.8	3.28	3.51	127.2
	G2	1.19	10.93	69.7	1.21	3.49	69.7	2.36	2.59	69.7
	A♯2	2.57	13.09	125.6	2.46	4.04	125.6	3.87	4.12	125.6
	C♯3	2.13	11.93	159.1	2.13	3.20	159.1	2.85	3.08	159.1
vlc	C♯2	0.83	11.25	107.9	0.70	2.68	107.9	1.96	2.20	107.9
	E2	0.80	10.53	71.2	0.75	2.12	71.2	1.22	1.45	71.2
	G2	2.79	12.53	96.8	2.83	4.85	96.8	2.37	2.60	98.9
	A♯2	0.82	10.55	144.5	0.72	3.01	155.7	2.07	2.30	162.5
	C♯3	1.34	12.32	106.0	1.20	3.17	110.9	2.51	2.76	110.9
Mean		1.00	7.41	118.9	0.94	2.25	119.9	1.52	1.67	121.9
Std. Dev.		0.66	2.69	45.2	0.65	1.11	45.6	0.80	0.84	46.2

Table B.2: Group 1, 3-oscillator, 2- and 1-table match optimization.

GROUP 1 4 Oscillators		Four-Table Match Optimization								
		4+0			2+2			3+1		
Instr	Pitch	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
bsn	A♯1	35.8	1682	99.7	31.6	162	101.9	26.3	145	99.2
	C♯2	40.7	1691	147.8	40.7	136	147.6	26.3	145	149.1
	E2	34.4	1681	98.1	21.8	99	98.1	23.7	141	101.2
	G2	54.3	1697	92.7	47.9	187	92.7	36.8	158	91.2
	A♯2	52.7	1696	74.1	47.3	159	74.2	47.6	168	76.1
	C♯3	44.5	1687	47.4	36.7	131	47.4	23.3	142	47.3
clb	C♯2	16.2	1660	178.5	16.9	84	191.1	19.3	135	179.2
	E2	83.6	1737	127.9	75.7	205	127.8	58.9	179	128.3
	G2	28.1	1671	145.8	31.3	135	142.9	17.1	135	143.1
	A♯2	51.9	1695	121.4	32.9	145	119.9	15.3	133	119.6
	C♯3	36.7	1680	101.6	28.5	116	101.9	21.1	140	101.8
hrn	E2	32.0	1675	102.7	22.5	117	102.1	18.8	136	101.3
	G2	61.1	1703	50.3	39.8	125	50.2	30.9	150	53.2
	A♯2	27.4	1669	67.5	17.7	103	67.5	19.6	137	67.9
	C♯3	79.2	1722	28.7	62.2	182	28.6	46.6	166	29.0
pno	A♯1	42.5	1686	60.6	35.9	130	60.2	22.0	140	61.2
	C♯2	77.1	1720	103.6	60.6	190	99.3	60.9	182	99.5
	E2	37.3	1679	140.4	28.1	139	147.1	15.2	133	143.7
	G2	41.2	1685	95.2	41.1	153	100.2	39.8	158	95.0
	A♯2	58.5	1701	119.5	23.5	136	129.2	19.7	139	118.3
	C♯3	37.7	1682	37.6	28.6	170	37.6	37.0	158	37.4
sax	A♯1	53.8	1693	73.1	36.0	173	73.5	39.8	159	74.0
	C♯2	50.9	1685	95.9	29.5	159	94.3	29.1	148	95.6
	E2	27.2	1663	114.7	27.7	139	114.6	22.1	140	118.8
	G2	54.2	1690	78.6	34.6	155	78.4	33.0	152	78.8
	A♯2	85.2	1720	72.9	70.7	209	72.5	48.7	168	73.2
	C♯3	42.3	1700	99.9	32.1	178	99.9	30.0	149	100.0
trb	E2	24.3	1695	67.2	17.5	102	67.2	20.1	138	67.2
	G2	27.1	1660	64.8	29.2	140	64.8	30.1	149	64.3
	A♯2	63.0	1696	50.9	60.6	199	50.9	33.0	153	51.9
	C♯3	40.5	1681	104.0	30.3	106	101.3	36.1	154	100.4
vla	C♯3	201.5	3844	242.9	126.3	569	243.1	151.1	434	238.4
vlb	A♯1	134.8	3319	84.7	100.6	506	84.7	81.4	330	85.5
	C♯2	147.9	3782	88.1	130.8	594	88.2	92.5	386	87.6
	E2	96.3	3279	107.3	75.6	532	107.4	109.3	361	107.3
	G2	191.8	3376	59.1	170.4	713	59.3	129.7	384	59.0
	A♯2	68.8	3512	97.6	59.8	406	97.6	40.5	304	97.3
	C♯3	45.6	3229	117.9	37.1	202	117.9	33.5	266	119.8
vlc	C♯2	102.6	3479	88.9	105.6	391	87.1	72.4	325	87.1
	E2	115.6	3298	62.7	111.3	464	62.7	108.8	357	63.4
	G2	117.3	3298	76.3	104.7	544	76.6	71.9	318	76.4
	A♯2	75.5	3258	90.5	65.2	230	97.3	48.4	273	94.2
	C♯3	135.2	3719	84.4	113.5	433	81.2	110.3	382	80.3
Mean		66.8	2181	94.5	54.4	236	95.1	46.5	203	94.5
Std. Dev.		43.5	807	38.5	36.5	165	39.5	33.5	92	38.1

Table B.3: Group 1, 4-oscillator, 4-table match optimization.

GROUP 1 4 Oscillators		Three-Table Match Optimization								
		3+0			2+1			1+2		
Instr	Pitch	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
bsn	A♯1	13.3	125	100.5	11.1	20.1	103.3	11.1	16.0	103.3
	C♯2	6.9	119	152.7	7.3	15.3	153.6	7.6	11.3	152.7
	E2	13.5	126	103.8	5.5	12.5	112.5	4.7	7.6	113.7
	G2	9.3	122	91.2	10.3	19.3	93.2	9.7	14.7	93.2
	A♯2	10.2	122	78.3	11.6	19.6	78.3	11.4	15.9	78.3
	C♯3	7.3	119	55.5	6.2	14.2	55.9	6.3	10.0	55.9
clb	C♯2	10.2	122	179.0	7.9	14.9	193.1	7.4	9.7	193.1
	E2	14.2	126	128.1	9.8	18.8	131.6	6.3	10.7	131.2
	G2	19.2	131	146.4	8.7	16.7	147.8	4.0	7.6	146.8
	A♯2	5.7	118	120.0	4.3	12.3	120.0	2.8	7.0	120.0
	C♯3	7.0	119	103.8	4.6	12.6	105.0	4.8	7.7	106.6
hrn	E2	6.5	119	103.3	6.2	14.2	103.3	6.2	10.3	103.3
	G2	8.7	121	56.4	6.1	14.1	54.5	6.2	9.6	54.5
	A♯2	2.6	115	69.5	2.6	10.6	69.9	2.8	6.8	69.4
	C♯3	42.1	154	28.5	38.6	47.6	28.5	41.9	46.7	28.5
pno	A♯1	14.4	126	61.6	8.2	16.2	63.4	8.2	11.9	63.4
	C♯2	22.6	135	99.8	13.3	22.3	100.2	8.7	13.5	100.3
	E2	18.6	131	148.0	16.9	24.9	148.0	18.8	24.3	148.2
	G2	14.2	126	97.8	7.4	16.4	108.8	4.0	8.7	110.5
	A♯2	6.1	118	132.8	4.8	13.8	132.0	7.8	13.3	132.4
	C♯3	9.2	121	38.4	7.9	16.9	38.4	8.0	13.4	38.4
sax	A♯1	9.4	121	74.6	6.2	15.2	75.2	6.3	11.7	75.2
	C♯2	10.9	123	97.6	5.7	14.7	100.1	4.8	9.7	100.1
	E2	7.4	119	119.8	4.6	13.6	120.7	4.2	8.6	120.7
	G2	10.0	122	81.1	9.2	18.2	81.1	6.3	10.7	80.9
	A♯2	6.8	119	75.0	5.3	14.3	74.4	5.5	11.1	74.4
	C♯3	8.1	120	102.7	8.7	18.7	101.4	8.5	14.6	103.2
trb	E2	10.9	123	66.9	9.3	17.3	66.9	8.8	12.2	66.9
	G2	8.4	120	66.5	6.2	15.2	67.2	6.2	10.9	67.2
	A♯2	5.8	118	53.4	5.7	14.7	53.1	5.7	11.5	53.1
	C♯3	9.8	122	118.5	22.4	29.4	96.4	20.1	23.3	95.6
vla	C♯3	21.3	272	243.5	20.9	46.9	245.7	22.4	39.7	244.7
vlb	A♯1	21.1	241	90.3	17.8	40.8	90.6	18.0	34.1	90.6
	C♯2	35.8	288	88.2	30.6	56.6	88.0	32.0	49.8	88.1
	E2	34.3	254	108.1	16.8	41.8	108.4	17.4	36.5	108.4
	G2	31.7	253	59.0	30.5	58.5	59.2	28.4	49.2	59.2
	A♯2	34.1	274	97.9	34.9	56.9	97.8	35.8	49.0	97.8
	C♯3	8.5	228	124.9	5.7	20.7	124.9	5.9	12.4	124.9
vlc	C♯2	21.4	252	87.1	19.4	39.4	88.9	17.2	28.0	88.9
	E2	34.7	252	62.9	34.2	55.2	62.9	34.3	47.6	62.9
	G2	23.4	240	75.2	23.0	47.0	75.4	23.4	40.0	75.4
	A♯2	29.6	247	94.1	28.0	43.0	94.6	50.3	58.4	101.0
	C♯3	27.5	271	81.9	25.2	46.2	81.5	21.3	34.7	81.5
Mean		15.6	160	96.9	13.2	25.5	97.6	13.3	20.5	97.8
Std. Dev.		10.2	61	38.9	9.8	15.1	39.9	11.5	15.1	39.8

Table B.4: Group 1, 4-oscillator, 3-table match optimization.

GROUP 1 5 Oscillators		Five-Table Match Optimization								
		4+1			3+2			2+2+1		
Instr	Pitch	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
bsn	A♯1	240	1895	89.2	255	547	79.9	258	398	83.3
	C♯2	221	1880	124.2	239	542	125.5	184	289	124.1
	E2	347	2002	89.5	292	541	87.3	237	321	88.7
	G2	224	1878	79.9	289	624	78.7	213	363	79.9
	A♯2	257	1910	63.3	214	537	63.8	290	412	63.0
	C♯3	336	1990	40.1	318	631	40.2	377	482	39.7
clb	C♯2	110	1759	166.4	101	320	167.4	88	161	168.5
	E2	210	1873	123.6	198	511	122.3	212	351	122.7
	G2	258	1909	129.5	306	566	130.5	292	404	130.8
	A♯2	222	1875	110.9	281	551	109.0	225	347	109.3
	C♯3	322	1974	93.2	373	664	92.2	370	466	93.1
hrn	E2	223	1874	82.0	107	345	81.5	215	317	81.3
	G2	173	1825	44.5	276	567	45.3	261	356	43.7
	A♯2	227	1877	54.4	140	389	53.2	187	279	54.1
	C♯3	162	1815	19.0	282	585	20.3	207	337	19.0
pno	A♯1	206	1858	52.1	287	569	50.1	213	317	51.4
	C♯2	365	2019	91.3	305	661	92.3	196	336	91.8
	E2	179	1829	105.6	216	487	118.2	266	385	119.0
	G2	302	1955	77.0	287	568	77.2	117	238	71.7
	A♯2	226	1879	100.8	274	566	94.0	327	448	109.7
	C♯3	281	1934	28.5	326	673	28.4	223	373	28.5
sax	A♯1	269	1918	63.3	263	566	64.6	365	512	65.9
	C♯2	344	1988	80.0	253	545	78.7	272	410	79.7
	E2	235	1878	106.6	236	496	101.8	218	336	106.4
	G2	320	1965	73.0	305	607	72.0	307	437	71.1
	A♯2	390	2036	63.1	214	526	63.3	363	512	64.4
	C♯3	312	1981	88.3	318	609	88.1	292	448	88.0
trb	E2	295	1975	60.0	314	584	60.5	296	389	60.0
	G2	203	1844	53.6	244	536	54.2	235	354	53.6
	A♯2	248	1892	41.8	250	564	44.0	266	415	41.8
	C♯3	208	1854	72.6	187	457	78.4	169	250	77.8
vla	C♯3	638	4333	217.2	643	1759	217.6	624	1119	216.3
vlb	A♯1	601	3826	71.7	649	1662	70.9	708	1154	71.7
	C♯2	592	4280	80.0	631	2020	79.7	577	1094	80.3
	E2	569	3788	94.4	526	1624	95.4	628	1121	94.4
	G2	418	3644	53.3	505	1585	53.1	470	1055	52.7
	A♯2	446	3924	82.8	654	1531	80.4	476	856	82.7
	C♯3	455	3663	95.4	738	1345	103.3	392	581	95.5
vlc	C♯2	810	4225	76.0	739	1573	76.8	727	1051	76.4
	E2	487	3710	58.4	360	1433	58.5	458	867	58.4
	G2	645	3868	60.8	589	1598	60.9	657	1136	60.4
	A♯2	731	3939	78.7	593	1033	85.7	460	648	79.3
	C♯3	777	4410	70.1	807	1803	71.1	722	1089	70.0
Mean		351	2483	81.5	358	835	81.8	340	540	81.9
Std. Dev.		178	947	35.2	184	491	35.4	170	303	35.7

Table B.5: Group 1, 5-oscillator, 5-table match optimization.

GROUP 1 5 Oscillators		Four-Table Match Optimization								
		4+0			2+2			3+1		
Instr	Pitch	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
bsn	A♯1	154	1800	90.1	134	264	90.4	111	230	92.6
	C♯2	185	1835	129.2	184	279	134.1	103	222	134.8
	E2	133	1780	90.5	83	160	90.5	92	209	94.4
	G2	246	1889	79.8	221	360	79.8	156	277	79.1
	A♯2	241	1884	62.6	230	342	62.5	232	352	63.0
	C♯3	192	1835	39.3	160	254	39.3	95	214	39.2
clb	C♯2	141	1785	166.0	108	175	171.2	183	299	166.9
	E2	192	1845	123.3	309	438	123.3	179	299	124.4
	G2	112	1755	129.6	104	208	131.4	59	177	134.1
	A♯2	153	1796	111.5	149	261	109.6	63	180	109.6
	C♯3	153	1796	94.2	118	205	95.0	82	201	93.9
hrn	E2	131	1774	86.1	87	181	85.8	42	159	89.2
	G2	246	1888	45.9	177	262	45.4	131	250	45.4
	A♯2	107	1749	54.6	64	149	55.1	340	457	54.3
	C♯3	252	1895	22.5	281	401	22.3	198	317	22.4
pno	A♯1	190	1833	52.0	159	253	52.0	87	205	51.9
	C♯2	298	1941	93.2	256	385	91.7	269	390	94.3
	E2	130	1772	117.2	135	246	124.9	185	303	119.8
	G2	178	1822	78.0	136	248	80.6	102	220	75.6
	A♯2	241	1884	104.3	90	203	112.4	83	202	105.8
	C♯3	159	1803	29.0	120	261	29.0	165	286	29.1
sax	A♯1	248	1887	62.4	152	289	65.8	177	296	62.8
	C♯2	191	1825	81.1	121	250	83.4	124	243	86.3
	E2	111	1747	108.9	124	235	109.3	88	206	110.1
	G2	171	1807	71.0	152	272	70.6	139	258	72.2
	A♯2	321	1956	63.7	210	348	62.7	233	352	63.1
	C♯3	172	1830	89.9	130	276	89.3	119	238	90.0
trb	E2	80	1751	58.5	61	145	58.5	78	196	58.5
	G2	110	1743	55.6	125	236	55.5	136	255	54.8
	A♯2	298	1931	42.1	302	440	41.8	142	262	44.9
	C♯3	229	1869	81.8	347	423	75.3	222	340	81.6
vla	C♯3	667	4309	218.0	622	1065	217.5	747	1030	219.9
vlb	A♯1	600	3784	73.0	478	883	72.5	372	621	72.0
	C♯2	690	4324	78.0	635	1098	78.1	414	708	77.9
	E2	354	3537	96.4	350	806	96.3	423	675	96.9
	G2	818	4002	52.3	787	1330	52.3	610	864	52.1
	A♯2	660	4103	80.0	614	960	80.0	438	701	79.8
	C♯3	149	3332	96.5	151	316	96.5	132	365	100.4
vlc	C♯2	402	3778	76.4	408	693	77.3	315	568	77.5
	E2	564	3746	57.5	530	883	57.6	520	768	58.7
	G2	510	3691	60.4	447	886	60.7	323	569	61.1
	A♯2	354	3536	74.9	283	448	80.7	190	415	78.9
	C♯3	686	4270	70.7	540	860	70.9	541	813	70.6
Mean		284	2398	82.5	253	434	83.2	220	377	83.5
Std. Dev.		194	942	35.7	183	306	36.4	164	218	36.3

Table B.6: Group 1, 5-oscillator, 4-table match optimization.

GROUP 1 5 Osc, 4 WT		Limiting Optimization Problem Size							
		Overlap=2		Limit=15		Limit=12		Overlap=1	
Instr	Pitch	T_{opt}	Error	T_{opt}	Error	T_{opt}	Error	T_{opt}	Error
bsn	A♯1	154	90.1	154	90.1	58.5	90.6	62.2	91.7
	C♯2	185	129.2	185	129.2	50.4	130.6	15.9	131.9
	E2	133	90.5	133	90.5	71.5	90.7	17.3	91.5
	G2	247	79.8	246	79.8	71.7	81.2	20.5	84.2
	A♯2	240	62.6	241	62.6	53.3	63.4	19.0	65.0
	C♯3	191	39.3	192	39.3	95.2	43.2	18.0	43.5
clb	C♯2	140	166.0	141	166.0	61.7	166.0	20.1	166.0
	E2	447	123.1	192	123.3	64.8	123.5	27.7	124.9
	G2	111	129.6	112	129.6	48.3	130.5	27.4	130.8
	A♯2	248	110.5	153	111.5	53.7	111.8	20.1	113.3
	C♯3	152	94.2	153	94.2	62.6	94.8	87.9	94.5
hrn	E2	131	86.1	131	86.1	64.0	86.2	46.5	84.3
	G2	291	45.9	246	45.9	49.3	46.7	21.3	46.7
	A♯2	106	54.6	107	54.6	55.7	55.5	153.4	54.9
	C♯3	421	22.4	252	22.5	49.8	23.8	27.3	24.7
pno	A♯1	191	52.0	190	52.0	68.8	53.1	74.1	53.0
	C♯2	391	93.2	298	93.2	50.3	93.5	32.2	93.7
	E2	177	117.1	130	117.2	45.8	117.2	86.8	119.6
	G2	176	78.0	178	78.0	63.2	79.4	18.9	80.7
	A♯2	280	102.8	241	104.3	73.2	121.5	26.5	122.6
	C♯3	158	29.0	159	29.0	61.3	29.0	20.3	29.3
sax	A♯1	246	62.4	248	62.4	67.9	64.0	18.4	65.8
	C♯2	233	81.1	191	81.1	99.9	83.7	24.6	85.3
	E2	110	108.9	111	108.9	44.4	108.9	53.4	107.0
	G2	252	71.0	171	71.0	72.7	73.1	21.0	74.3
	A♯2	431	62.3	321	63.7	41.4	64.3	30.5	64.4
	C♯3	170	89.9	172	89.9	81.0	90.4	18.6	92.0
trb	E2	78	58.5	80	58.5	49.5	58.6	9.7	58.9
	G2	108	55.6	110	55.6	45.6	56.1	12.3	57.7
	A♯2	293	42.1	298	42.1	50.9	46.2	20.1	46.4
	C♯3	334	81.6	229	81.8	84.0	84.7	34.4	86.9
vla	C♯3	1050	217.4	667	218.0	129.7	222.6	65.2	223.6
vlb	A♯1	650	72.7	600	73.0	129.7	74.1	55.1	74.5
	C♯2	684	78.0	690	78.0	137.0	79.0	57.1	79.0
	E2	440	96.4	354	96.4	119.2	96.6	37.1	97.4
	G2	1024	52.1	818	52.3	105.1	53.0	84.3	53.1
	A♯2	735	80.0	660	80.0	410.3	80.1	136.3	81.3
	C♯3	207	96.5	149	96.5	65.0	96.6	25.3	99.3
vlc	C♯2	487	75.7	402	76.4	200.6	76.5	62.3	76.8
	E2	563	57.5	564	57.5	103.7	58.3	46.2	58.6
	G2	597	60.4	510	60.4	116.4	61.0	65.6	62.3
	A♯2	359	74.9	354	74.9	139.1	77.6	59.3	80.7
	C♯3	682	70.7	686	70.7	159.6	71.3	69.3	72.2
Mean		333	82.4	284	82.5	86.6	83.9	43.0	84.8
Std. Dev.		238	35.6	194	35.7	61.9	36.1	31.9	36.2

Table B.7: Optimization time and RMS error of Group 1, 5-oscillator, 4-table matches with limits on overlapping and wavetable set size.

GROUP 1 5 Osc, 4 WT		Optimization Graph Size (in thousands)							
		Overlap=2		Limit=15		Limit=12		Overlap=1	
Instr	Pitch	V	E	V	E	V	E	V	E
bsn	A♯1	29.1	1815	29.1	1815	14.7	744	16.0	769
	C♯2	34.6	2112	34.6	2112	13.3	636	5.9	196
	E2	28.5	1577	28.5	1577	17.5	883	6.1	221
	G2	44.8	2855	44.8	2855	17.4	878	7.4	252
	A♯2	45.1	2757	45.1	2757	14.9	651	6.8	227
	C♯3	37.4	2248	37.4	2248	21.8	1175	6.8	220
clb	C♯2	24.8	1613	24.8	1613	13.7	747	6.2	235
	E2	75.2	4879	40.1	2191	17.3	795	9.6	326
	G2	22.9	1338	22.9	1338	12.9	610	8.3	328
	A♯2	46.5	2801	32.4	1790	15.2	664	7.1	240
	C♯3	31.1	1803	31.1	1803	16.5	769	19.3	993
hrn	E2	26.7	1551	26.7	1551	15.4	800	12.6	560
	G2	53.4	3277	47.0	2771	13.8	615	7.6	261
	A♯2	22.6	1294	22.6	1294	14.4	710	30.1	1742
	C♯3	69.8	4627	49.7	2808	14.5	613	9.3	327
pno	A♯1	34.9	2237	34.9	2237	17.5	858	17.6	869
	C♯2	67.3	4384	54.9	3389	14.4	619	10.3	387
	E2	32.2	1994	26.3	1477	12.2	578	19.5	986
	G2	33.4	2092	33.4	2092	16.1	789	6.7	232
	A♯2	50.0	3222	43.9	2778	18.3	910	9.0	320
	C♯3	31.6	1865	31.6	1865	15.6	760	6.7	251
sax	A♯1	45.5	2823	45.5	2823	17.5	833	6.8	220
	C♯2	42.8	2703	36.9	2223	22.9	1225	8.2	302
	E2	22.4	1304	22.4	1304	11.7	570	13.9	615
	G2	47.8	2882	36.0	1989	18.4	904	7.6	254
	A♯2	75.8	4789	60.3	3600	12.4	513	10.1	370
	C♯3	34.0	2040	34.0	2040	19.6	1005	6.6	234
trb	E2	17.7	979	17.7	979	13.5	631	3.9	122
	G2	22.2	1300	22.2	1300	12.1	576	4.8	153
	A♯2	53.8	3308	53.8	3308	14.3	621	7.3	240
	C♯3	54.7	3692	41.2	2609	19.5	1043	10.3	422
vla	C♯3	174.2	11649	121.9	7471	35.8	1584	22.1	782
vlb	A♯1	114.3	7360	106.9	6760	34.1	1572	18.0	673
	C♯2	124.5	7812	124.5	7812	36.7	1670	19.3	697
	E2	82.0	4976	70.3	4056	31.3	1460	13.1	447
	G2	165.6	11334	139.9	9115	30.5	1277	25.3	1028
	A♯2	120.9	8346	111.1	7520	76.9	4895	33.1	1640
	C♯3	41.1	2399	32.7	1783	17.8	853	9.1	323
vlc	C♯2	89.4	5685	77.0	4753	44.7	2485	19.1	800
	E2	102.2	6431	102.2	6431	28.5	1262	15.7	571
	G2	104.6	6817	92.4	5878	30.8	1451	19.6	817
	A♯2	63.9	4197	63.9	4197	32.2	1778	17.4	779
	C♯3	117.0	7783	117.0	7783	39.7	1941	21.4	856
Mean		59.5	3790	52.8	3258	21.6	1069	12.6	518
Std. Dev.		38.8	2641	33.0	2175	12.2	740	7.1	373

Table B.8: Numbers (in thousands) of vertices and edges in the optimization graphs for Group 1, 5-oscillator, 4-table matches with limits on overlapping and wavetable set size.

GROUP 1 $N_{osc} = N_{WT}$		Augmented GA (Thorough) Match Optimization								
		2+1			3+1			4+1		
Instr	Pitch	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
bsn	A♯1	2.69	48.0	114.7	24.2	118	99.2	215	385	91.4
	C♯2	1.54	57.3	171.7	42.2	162	146.0	161	336	126.5
	E2	1.30	43.7	118.3	17.6	113	104.1	196	375	91.7
	G2	1.67	38.0	104.7	39.6	158	93.6	384	553	81.5
	A♯2	2.46	59.8	90.1	39.0	157	74.9	270	467	66.5
	C♯3	1.67	43.0	70.2	26.8	124	49.4	411	614	41.3
clb	C♯2	0.50	41.7	206.8	16.7	120	193.2	205	336	182.8
	E2	2.59	41.3	134.9	37.3	119	127.5	211	317	122.8
	G2	1.84	64.2	160.2	19.7	126	142.9	216	383	131.6
	A♯2	0.87	57.0	135.3	19.1	110	120.0	274	428	109.4
	C♯3	1.45	56.5	118.1	24.6	142	102.2	177	302	93.8
hrn	E2	0.79	49.9	121.8	6.3	137	102.0	209	383	91.7
	G2	1.17	69.2	65.6	31.6	133	53.1	287	497	43.3
	A♯2	0.85	58.1	86.6	11.4	128	67.8	143	310	55.7
	C♯3	2.19	57.5	39.1	33.6	120	30.1	359	550	24.8
pno	A♯1	1.29	67.5	75.9	12.4	114	62.8	220	368	53.7
	C♯2	2.00	37.5	107.8	52.5	142	100.2	277	391	94.2
	E2	3.09	43.0	185.7	57.6	130	146.6	148	274	127.7
	G2	2.32	26.0	117.1	18.4	100	94.8	124	297	75.3
	A♯2	1.46	50.0	155.8	20.6	116	131.0	288	419	102.1
	C♯3	1.50	70.1	49.3	50.2	120	39.8	289	516	29.6
sax	A♯1	2.04	52.1	81.3	48.1	161	74.0	232	370	63.3
	C♯2	1.54	55.7	108.8	26.4	102	96.3	284	449	86.1
	E2	1.57	39.4	129.2	23.9	102	118.9	235	368	113.7
	G2	1.51	50.9	96.6	37.9	140	81.1	367	522	71.2
	A♯2	1.46	17.6	85.5	46.5	150	73.2	294	484	63.3
	C♯3	1.92	53.7	120.6	49.2	115	100.2	270	436	90.8
trb	E2	0.91	54.1	79.2	12.8	140	67.3	191	374	61.0
	G2	1.74	55.5	82.7	11.5	142	65.4	233	456	56.5
	A♯2	2.11	47.8	74.0	30.4	109	52.7	309	495	43.6
	C♯3	1.58	39.2	141.5	34.4	142	108.4	226	376	78.7
vla	C♯3	4.59	262.3	267.8	110.5	463	243.5	526	922	225.7
vlb	A♯1	4.56	205.1	107.7	82.2	429	86.5	462	886	72.3
	C♯2	7.47	232.2	103.0	102.5	562	88.4	570	1418	81.0
	E2	5.09	116.8	121.8	109.5	341	108.6	481	885	99.5
	G2	6.27	222.1	69.4	131.1	493	59.1	570	1028	53.4
	A♯2	2.50	295.4	126.9	59.9	464	102.6	510	1236	82.4
	C♯3	1.88	197.4	162.6	35.4	404	123.5	544	1163	100.9
vlc	C♯2	4.17	207.3	106.9	64.5	438	86.8	461	910	76.5
	E2	6.94	170.7	71.2	121.8	512	63.0	624	1074	58.7
	G2	4.70	198.1	96.7	73.5	450	73.9	816	1358	60.3
	A♯2	3.10	171.8	126.5	26.7	286	95.1	699	1204	80.4
	C♯3	5.62	202.8	94.5	74.8	565	79.3	579	1284	70.9
Mean		2.52	93.7	113.6	44.5	219	96.0	338	609	84.4
Std. Dev.		1.72	75.6	42.8	31.7	153	39.4	166	340	37.4

Table B.9: Optimization of matches found by augmented thorough GA search. The "2+1," "3+1," and "4+1" matches are optimized for 3, 4, and 5 oscillators, respectively.

GROUP 1 $N_{osc} = N_{WT}$		Augmented GA (Quick) Match Optimization								
		2+1			3+1			4+1		
Instr	Pitch	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error	T_{opt}	T_{total}	Error
bsn	A♯1	2.90	22.1	114.5	39.6	90.3	99.9	219	257	91.8
	C♯2	1.48	21.0	171.2	25.5	58.1	150.9	202	393	126.8
	E2	1.36	21.2	118.3	16.5	50.0	105.9	292	338	94.2
	G2	2.11	21.1	105.1	29.5	70.5	91.8	315	370	79.3
	A♯2	2.51	12.5	97.8	59.0	95.0	79.8	200	241	66.2
	C♯3	1.65	11.1	79.2	26.4	50.6	49.3	320	365	40.0
clb	C♯2	0.74	20.7	214.2	19.2	58.8	196.9	120	178	171.4
	E2	2.25	15.8	138.3	44.1	79.2	129.2	252	326	122.6
	G2	2.08	16.5	162.0	31.9	62.4	148.5	256	303	136.6
	A♯2	1.14	16.9	144.4	29.1	54.0	124.0	235	283	108.8
	C♯3	1.52	16.9	118.5	22.3	58.9	105.0	203	419	93.7
hrn	E2	1.04	20.7	122.1	13.7	50.5	103.2	171	218	98.3
	G2	1.25	20.4	65.6	17.4	53.6	49.8	359	402	44.6
	A♯2	1.01	21.5	88.8	6.7	41.7	64.5	161	237	58.9
	C♯3	2.10	25.4	38.9	41.7	73.4	29.4	261	300	24.7
pno	A♯1	1.46	15.6	75.7	25.7	51.0	65.0	203	239	55.4
	C♯2	2.35	19.2	106.7	60.8	93.3	99.8	240	304	94.8
	E2	1.49	19.1	199.4	21.4	62.7	161.0	138	175	125.5
	G2	1.83	13.8	118.3	20.0	35.8	109.0	123	160	85.2
	A♯2	1.28	17.8	156.3	10.8	34.1	131.4	300	340	112.4
	C♯3	1.69	20.8	49.6	52.6	83.0	40.0	335	385	31.7
sax	A♯1	1.72	16.0	84.5	45.1	82.7	73.9	256	295	68.0
	C♯2	1.47	18.0	109.3	28.0	70.1	101.2	283	342	85.0
	E2	1.66	14.3	131.2	32.5	58.0	124.0	158	216	110.8
	G2	1.59	20.8	95.4	29.7	53.9	83.9	287	335	71.5
	A♯2	1.61	17.2	83.6	37.8	70.9	74.3	261	307	62.5
	C♯3	2.51	23.3	118.2	34.3	57.8	103.9	273	471	92.7
trb	E2	1.27	17.2	86.1	20.8	47.9	69.9	191	266	60.5
	G2	1.48	22.4	83.4	29.8	51.2	70.7	231	269	60.3
	A♯2	1.69	23.3	65.0	27.4	73.4	52.6	238	291	46.9
	C♯3	1.30	15.3	142.5	9.8	45.0	97.1	104	142	82.4
vla	C♯3	5.74	73.2	269.1	92.1	209.5	242.1	806	1045	221.3
vlb	A♯1	4.56	55.0	107.4	82.0	176.3	88.5	559	696	77.5
	C♯2	6.50	85.4	103.4	130.3	245.2	88.4	761	961	80.4
	E2	5.29	50.3	124.5	102.6	190.1	108.4	539	697	101.0
	G2	7.13	77.2	69.2	169.4	287.0	59.4	595	724	54.3
	A♯2	2.14	67.8	125.8	71.5	182.5	108.0	497	735	90.7
	C♯3	2.30	50.0	171.4	53.1	151.7	127.9	168	317	99.8
vlc	C♯2	5.12	57.3	105.3	72.9	205.7	86.9	771	916	80.3
	E2	5.62	58.1	73.1	123.0	211.2	64.0	523	982	59.1
	G2	4.52	56.8	97.5	117.2	226.8	76.8	611	739	62.6
	A♯2	1.08	53.4	119.3	69.7	191.3	109.3	585	761	83.7
	C♯3	6.03	74.6	98.6	91.0	238.0	82.1	510	704	73.2
Mean		2.50	31.1	115.1	48.5	103.1	98.3	328	429	85.8
Std. Dev.		1.72	21.3	44.1	37.1	70.9	40.5	187	247	36.1

Table B.10: Optimization of matches found by augmented quick GA search. The "2+1," "3+1," and "4+1" matches are optimized for 3, 4, and 5 oscillators, respectively.

GROUP 1		Horner's Constrained Matching					
		3 Oscillators		4 Oscillators		5 Oscillators	
Instr	Pitch	Time	Error	Time	Error	Time	Error
bsn	A♯1	1.06	125.0	10.8	113.3	158	94.2
	C♯2	1.06	191.3	10.7	161.2	155	137.8
	E2	1.06	140.9	10.7	126.4	155	111.2
	G2	1.06	122.2	10.7	98.0	157	91.5
	A♯2	1.06	113.5	10.7	82.8	156	70.7
	C♯3	1.06	101.2	10.7	64.4	154	52.5
clb	C♯2	1.06	232.5	10.7	202.1	154	174.6
	E2	1.07	144.1	10.7	132.7	154	125.8
	G2	1.01	182.4	10.7	158.8	154	142.5
	A♯2	1.17	152.7	10.7	127.6	154	119.1
	C♯3	1.09	146.9	10.7	125.6	154	113.3
hrn	E2	1.08	187.8	10.7	112.2	154	96.2
	G2	1.13	81.6	10.7	60.5	154	48.0
	A♯2	1.14	109.8	10.7	76.5	154	62.1
	C♯3	1.06	48.6	10.7	31.8	154	25.3
pno	A♯1	1.05	92.0	10.7	71.0	154	59.1
	C♯2	1.06	110.9	10.7	102.0	154	97.2
	E2	1.06	217.3	10.7	170.2	154	121.4
	G2	1.06	169.1	10.7	99.9	154	87.7
	A♯2	1.15	176.8	10.7	145.5	154	109.9
	C♯3	1.03	62.8	10.7	42.7	154	33.6
sax	A♯1	1.09	86.6	10.7	77.1	154	66.7
	C♯2	1.05	119.9	10.7	106.4	154	93.3
	E2	1.06	138.9	10.7	128.8	154	121.9
	G2	1.07	110.5	10.7	89.5	154	81.1
	A♯2	1.06	90.0	10.7	78.5	154	69.5
	C♯3	1.06	135.6	10.7	112.4	154	100.1
trb	E2	1.06	96.1	10.7	73.8	154	60.8
	G2	1.06	106.7	10.7	80.3	154	64.9
	A♯2	1.06	90.2	10.7	66.1	154	51.6
	C♯3	1.07	164.9	10.7	135.2	154	96.1
vla	C♯3	2.02	299.8	13.1	254.9	159	230.9
vlb	A♯1	1.78	126.5	12.5	96.1	159	76.9
	C♯2	2.03	127.5	13.1	101.4	161	81.7
	E2	1.80	131.1	12.5	113.7	160	105.4
	G2	1.80	72.3	12.5	61.5	160	54.9
	A♯2	1.87	166.0	12.8	117.5	158	92.7
	C♯3	1.78	252.0	12.5	136.5	159	111.3
vlc	C♯2	1.84	119.0	12.7	94.8	159	82.1
	E2	1.79	74.7	12.5	67.2	159	61.0
	G2	1.78	104.3	12.5	84.8	159	65.2
	A♯2	1.77	184.5	12.5	110.6	158	87.8
	C♯3	1.97	147.7	13.0	94.2	163	80.4
Mean		1.29	136.1	11.3	106.7	156	90.9
Std. Dev.		0.36	51.7	0.9	41.7	3	37.2

Table B.11: Group 1, Horner's constrained matching results.